WEB-BASED DATA PROCESSING FOR ENVIRONMENTAL SURVEILLANCE MONITORING APPLICATIONS

By

Lola Xiomara Bautista Rozo

A thesis submitted in partial fulfillment of the requirements for the degree of
MASTER IN SCIENCE
in
COMPUTER ENGINEERING
(Digital Signal Processing)

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
May, 2007

Approved by:

Nestor J. Rodriguez, PhD. Member, Graduate Committee	Date
Manuel Rodriguez, PhD. Member, Graduate Committee	Date
Domingo Rodriguez, PhD. President, Graduate Committee	Date
Dorial Castellanos, PhD. Representative of Graduate Studies	Date
Isidoro Couvertier, PhD. Chairperson of the Department	Date

ABSTRACT

This work deals with the development of a Java-based environment for the treatment of remote sensing imaging information on a cyber-infrastructure. Especial attention is given to the development of a computational signal algebra framework for the modeling and simulation of digital image interferometry processing applications.

Correlated digital interferometry (CDI) for imaging radars deals with the use of signal correlation techniques to process the phase information of digital image representations of microwave imaging signals.

A huge productivity advantage would be possible if applications written in scripting languages, such as $MATLAB^{®}$, could be compiled into highly optimized machine code (i.e. C, Java). It implies a library preprocessing phase to extensively analyze and optimize collections of libraries that define an extendend language.

RESUMEN

Este trabajo se centra en el desarrollo de un ambiente basado en Java para el tratamiento de imágenes adquiridas por la técnica de percepción remota sobre una ciber-infraestructura. Se ha dado especial atencion al desarrollo de un marco computacional de álgebra de señales para el modelamiento y simulación de aplicaciones de procesamiento de interferometría en imágenes digitales.

La interferometría digital correlacionada (CDI por sus siglas en inglés), tiene que ver con el uso de técnicas de correlación de señales para procesar la información de la fase de las representaciones de imágenes digitales de señales de microondas.

Una alta ventaja en productividad es posible si aplicaciones escritas en lenguajes "script", tales como $MATLAB^{®}$, pueden ser compilados dentro de código máquina altamente optimizado (como por ejemplo C y Java). Esto implica una fase de preprocesamiento de librerías para analizar extensivamente y optimizar colecciones de librerías que han sido definidas en un lenguaje.

by

Lola Xiomara Bautista Rozo

Dedicated to

my mother María Elena,

my father Libardo,

my brother Sergio,

my sister Norma, and

to my aunt Flor Elva, in loving memory.

ACKNOWLEDGMENTS

I would like to thank very specially to my advisor Dr. Domingo Rodríguez for giving me the opportunity to work with him, for his continuous support, dedication and help in this project, and for his friendship during all these years. Also, I would like to thank you to my graduate committee formed by Dr. Manuel Rodriguez and Dr. Nestor Rodriguez for their contributions to my work.

This is the opportunity to thank to all people that were with me in this stage of my life. Anita, Carlitos, Carito and Mariana who were the best housemates. John for his unconditional friendship. Jorge, Jazz, Mariana and Valeria for give me a place in their home and their love. Cesar, Sandra and Cata for be a lovely family. Andrea, Tatty and Edwin for be the friends of always. Checho, Fabian, Felix, Aponte and Diego for their support and shared moments. Victor, Karla and Feisar for their constant support in the distance.

This work was supported by National Science Foundation (NSF) under CISE_CNS grant No. 0424546.

Contents

List of Figures

Chapter 1

INTRODUCTION

The activity of monitoring the environment have increased for the last 40 years since the National Oceanic and Atmospheric Administration (NOAA) launched the world's first weather satellite [1], the National Aeronautics and Space Administration (NASA) started their TIROS program (Television Infrared Observation Satellite), to determine if satellites could be useful in the study of the Earth [2]. Another agencies like the Department of Natural Resources (DNR), and the Environmental Protection Agency (EPA) also are contributing to the development of programs to monitor many aspects of the environment like atmosphere, oceans, land and organisms life, using different kind of sensors to acquire raw data in forms of time-series and digital images.

This work focused on web-based information processing pertaining to water-related ecological applications, or hydro-ecological applications. Hydrological monitoring is very useful to determining the water balance of a region, predicting flood, landslide and drought risk, designing bridges, predicting erosion or sedimentation, and establishing environmental policy guidelines [3]. Data acquired from monitoring exhibit to both spatial and temporal dimensions since most of the variables measured (topography, soils, vegetal covers, etc) are

distributed spatially and these variables present time variance and evolution that need to be tracked [4].

Research of the environment and the Earth involves several interdisciplinary fields, in order to establish the states of the ocean, atmosphere or climate: *Meteorology*, which describes, explains, and predicts the weather based on the interaction of principally the ocean and atmosphere. *Climatology*, which describes and explains the climate in terms of the interaction of the spheres of the Earth. *Atmospheric chemistry*, which describes, explains and predict the chemical composition of the atmosphere in principally terms of the interactions of the ocean, atmosphere, biosphere and human influence. *Hydrology*, which considers the flow of water through the Earth, from the transition of water in the form of precipitation in the atmosphere, to rivers, and groundwater in aquifers. [3]

Data used for monitoring analysis comes from two main sources: In-situ sensors and remote sensors. In-situ sensors measure a physical property within the area immediately sorrounding the sensor, while remote sensors measure physical properties at some distance from the sensor [5]. The quantity of data depends on the sampling frequency established in time and space. In-situ sensors usually collects data through time. This process is well known as a time-series. A large amount of data acquired with remote sensors is generally represented as digital images, which can be treated with digital image processing techniques. Time-series analysis has two objectives, identify the nature of the phenomenon and forecasting (predict future values) [6]. Imaging remote sensing has the potential to observe areas rather than merely points, and is possible to quantify precipitation, rainfall, and soil moisture [4].

Large sets of data are available in databases from many agencies and research groups that

work with imaging applications. Those databases can be found in web sites, which allow users make queries to retrieve that data. Geographical Information Systems (GIS) have very useful tools to display data in different layers that help users to attain better visualization, storage, and processing.

Web-based applications are those that work in a web server and can be accessed through a web browser over a network environment. These web-based applications technologies are enhancing the manner in which information is shared among signal processing nodes, stations, or stages. The enhancement comes about from the advantages of these technologies which support distribution of data, ease of navigation through different kinds of browsers as well as operating systems, and improved automated installation process.

This document presents a description of the development of a web-based environment for the treatment of one-dimensional and two-dimensional data. The environment is part of a new conceptual framework for the automated processing of information arriving from physical sensors in a generalized wide-area, large-scale distributed network infrastructure [7].

The project is constructing a Computational and Information Processing (CIP) environment to deal with the algorithmic treatment of signal-based large scale content in order to extract information relevant to a user (See Figure 1.1). Treating signals as elements in prescribed sets allows to study structures associated with such sets and to apply signal operator theoretic methods in a generalized computation and information setting.

The web-based environment implements a Computational Signal Processing System (CSPS)(red box in Figure 1.1), which contains operator methods, implemented through algorithmic formulations, to assist at improving and enhancing the performance in effecting some of the

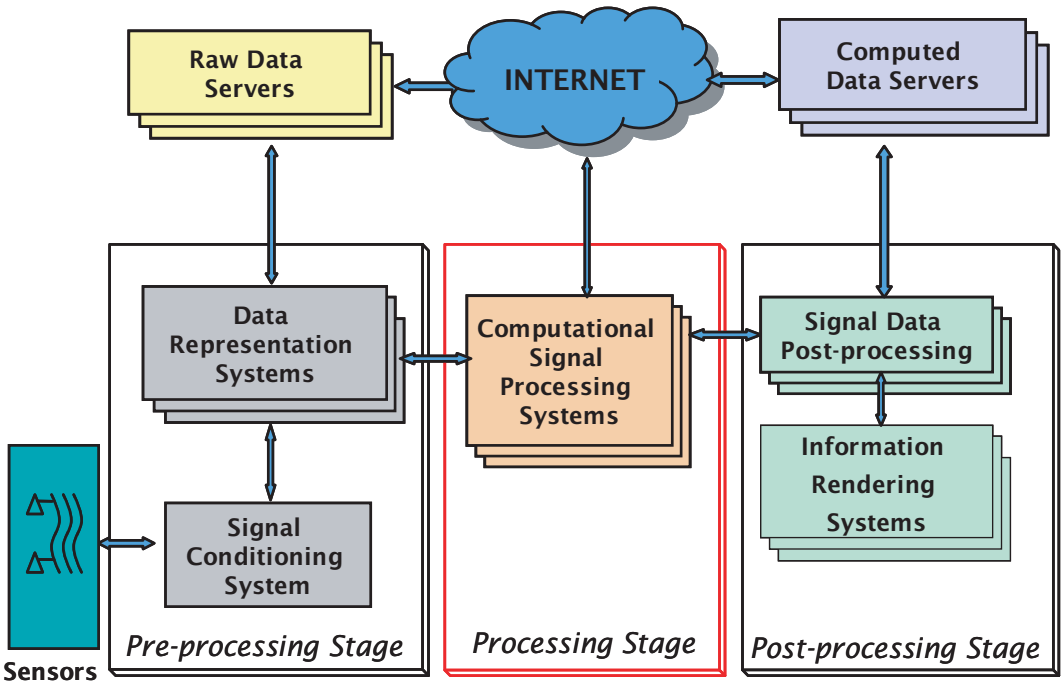

Figure 1.1: Conceptual Diagram of a Computational and Information Processing Environment

analysis tasks. The implementation of the CSPS was done using the Java 2 Platform, Enterprise Edition (J2EE) of Java, which offers an structured developing of multitier applications. In order to enhance the prototyping of operators was taken into account the benefits that offer the $MATLAB^{®}$ Compiler, which allows convert $MATLAB^{®}$ scripts as Java classes. This is described in more detail in Chapter 3.

The next sections of this chapter present the justification and justification and objectives of this research, as well as the original contributions that were reached. Chapter 2 describes in a detailed manner the mathematical background that is necessary to understand the computational signal processing environment. The computational implementation of the CSPS is explained in Chapter 4, and the results obtained from its application are presented en Chapter 5.

1.1 Previous Work

The literature review and analysis of previous works that appears in this section is organized in three main topics: works related to environmental monitoring, computational signal processing, and web-based applications.

Environmental hazards occur through the combination of natural forces and the human intervention in ecosystems. For example, environmental hydrology deals with the understanding of the behavior of natural forces, and the understanding of the effects caused by people. Alecu and Stancalie [8] used geographic information and remotely-sensed data to deals with the problem of forecasting and monitoring of torrential precipitation. They found a methodology that allow to determine flooding risk, taking measures from some points of the entire territory of study. Bryant and Rainey [9] used another technique to analyze data of flood inundation; they made use of time-series of AVHRR (Advanced Very High Resolution Radiometer) data. In this study they were looking for methods to compare hydrological response to seasonal changes. Frappart et. al [10] combined microwave data and in-situ data from hydrographic stations to determine the temporal variations of water volume stored in the floodplains of great river basins.

With respect to the technology used to transmit and process acquired data, many works have been done; however here are presented two that have a seamless approach to the work of this document. Chuan and Lim [11] performed tests of processing time using a client-server approach to process AVHRR data of forest fire, and Crowley [12] used wireless sensor networks to transmit data of temperature; the information was sent to a server via a GSM network.

From the mathematical perspective, there is an important work made by Ritter [13]

called *Image Algebra*. This Image Algebra is a mathematical theory concerned with the transformation and analysis of images. It was created due to the need for a common image-processing language, where could be together operands and operators. As a result of his work, it was developed a C++ library with the structures representing operands and operators. This library has been used specially for hardware implementations of specialized applications.

Angus [14] gave an object-oriented approach to that Image Algebra. In this work was established the abstract data types that represent images. Lenz et. al [15] also presents a work where they made use of group representation theory to design filter systems that analyze multichannel images, and as result proposed an algorithm that could be used in math-programs as $MATLAB^{®}$.

In the context of software systems to process data, there exist projects in the areas of environmental monitoring and management of medical data, constructed by using web application frameworks and grid computing. Kiehle [16] presents a technical solution of how to use web services technology for distributed storage and access of data using Open Geospatial Consortium (OGC) specifications.

References [17] and [18] worked together developing the Earth System Modeling Framework. In the project were built open-source standards to increase easy distribution of code and interoperability of components. Main targets of applications were in fields of the Earth sciences. Diviacco [19] also was working in the development of web-based open source solutions for seismic data.

Indiana University [20] had developed a metadata catalog called "MyLEAD". It stores the metadata associated with data products generated and used in the course of scientific

investigation. MyLEAD was built as a Web Service. This tool can easily integrate scientific databases and other data resources on the grid.

Nogueras-Iso et al. [21] illustrate the benefits of using standard catalogs in the development of Spatial Data Infrastructure and how they can be used for building other kind of specific services. Spatial Data Infrastructures (SDI) provide the framework for the optimization of the creation, maintenance and distribution of geographic information at different organization levels (regional, national or global). They identified that one of the main components of an SDI is a geographic catalog that enables users, or application software, to find the information that already exists within a distributed computing environment.

The SAR Atlas (SARA) project [22] developed by the Center for Advanced Computing Research (CACR) of the California Institute of Technology wants to demonstrate the serving of remote-sensing data across a gigabit wide-area network using a Netscape front end. This can process multispectral SAR data and the client communicates with a Web server, which knows metadata about the SAR images, such as the position of the image on the surface of the Earth.

The Geosciences Network (GEON) [23] is developing a cyber-infrastructure for integrative research to enable transformative advances in Geoscience research and education. GEON is "inherently a distributed system, since the scientists-who are users as well as providers of resources (e.g., data, tools, computing and visualization capabilities)-are themselves distributed".

Another application that is able in Internet is InfoVis Cyber-infrastructure, a Data-Code-Compute for Research and Education in Information Visualization [24]. It provides a unified architecture in which diverse data analysis, modeling and visualization algorithms can be plugged in and run.

In [25] and [26], Beer et. al. describe the evolution of a Java-based visual information system for image reconstruction algorithms, from a stand-alone application to a distributed application with the option of binding XML. Matsopoulos [27] et al, also performed a work of relating obstetrical, gynecological, and radiological data in a distributed system that provides tools for image enhancement, image segmentation, image classification and registration, in order to provide to the medical experts of a unified patient management mechanism.

Page et al. [28] presents "a programmable Java Distributed System, which utilizes the free resources of a heterogeneous set of computers linked together by a network". One of the important topics to highlight of this paper is the development of a "fully cross-platform parallel database search program" which permits searching information in a dynamic way over the nodes machines. This paper serves as a guide for the design of the main components of distributed systems, such as the servers, the clients and the interfaces and communications between them with the applications.

Manolakos et al. [29] gives special attention to the issue of exploiting the capability of $MATLAB^{®}$ to incorporate Java code for the "development of parallel and distributed component-based applications for heterogeneous clusters of workstations". In this paper is possible to see how $MATLAB^{®}$ can interacts with Java classes, specially because most of the algorithms for signal and image processing have been implemented already as $MATLAB^{®}$'s functions and avoid to rewrite code in Java.

Huallparimachi [30] in his master thesis designed and developed a Java-based distributed system tool-environment for image analysis with special attention given to synthetic aperture radar (SAR) imaging applications. This work has been taken as the initial point for our research, by studying the functionalities that were implemented to include the necessary

operations to perform correlated digital interferometry.

References [31] and [32] both are related to the implementation of educational tools for the learning of digital signal processing and image processing concepts. The Java-based tool developed by Campbel et. al. [31] was used in the course of Image Processing and Pattern Recognition at the Queen's University Belfast. Spanias and Bizuneh [32] developed an on-line DSP laboratory at the Arizona State University, which is used in the courses of DSP, Image Processing, Wireless Communications and Control.

1.2 Justification

As can be seen in the previous work section, digital signal processing and image processing are techniques used in different fields, as hydrology and medicine; however although the conditions of acquisition of data are completely different, most of the final outputs displayed to a user, are relatively seamless. Also, most of the techniques used to make processing in one of them, is also used in the other, for example, applying filters for image enhancement is a common tool that is available in specialized software for either hydrology and medicine.

The advantage to create a Computational Signal Processing System that deals with the algorithmic treatment of signal-based data, is that whatever be the kind of data, or whatever be its source, this can be processed by standard methods established with a formal mathematical notation, which can be translated to any programming language. As was done by Ritter [13] who established an image algebra to define a common language for the treatment of images, in this thesis work, one of the interest was to make an approach to obtain a common language for multidimensional signals, although the application is more related just to one-dimensional and two-dimensional signals.

1.3 Thesis Objectives

The thesis objectives are as follows:

Develop a web-based environment for the treatment of one-dimensional and two-dimensional signals, through the definition of a Computational Signal Processing System (CSPS). These imply the achievement of the following specific objectives:

- Define the main components of a CSPS:
  - A set of input entities
  - A set of output entities
  - A set of generalized computational and information processing algorithms
  - A set of action rules for these operators
  - A set of action rules for the operators to act on input entities
  - A user interface
- Implement features of the signal algebra framework using $MATLAB^{®}$ and Java language.

1.4 Research Methodology

In this section is described the tasks needed to achieve the objectives showed before. This is divided in three phases: Beginning, Construction, and Transition.

Phase 1: Beginning

This phase was made the preliminary searching of information needed to understand the problem. As final result of this phase, we will have a list of system requirements that will be implemented in the next phase. The following are the tasks of this phase:

- Collect the basic requirements of potential users of the tool, in order to clarify the required functionality of the system.
- Make an initial study of the theory related with the mathematical framework of the signal algebra. In this way it is possible to establish the algorithms that will be implemented.
- Make a deep study of the tools that will be used for implementation, such as $MATLAB^{®}$ and Java, specially, to analyze its interaction.
- Study the concepts related with web applications and frameworks to develop it.
- Search related works in the areas of image processing, signal algebra and web applications.

Phase 2: Construction

The objective of this phase is to perform the tasks related with implementation of mathematical algorithms and developing of the system. These are the tasks:

- Implement the mathematical algorithms of the Computational Signal Processing System.
- Design and implement the architecture of a web application.
- Implement the functionalities identified in the beginning phase.
- Develop the user interface using Usability Engineering and Human Computer Interaction.

Phase 3: Transition

In this phase the work will be concentrated on performing the necessary proves of the system in order to correct faults and errors detected in the algorithms. Also, will be performed the Usability tests to improve the user interface.

1.5 Original Contributions

1. Conceptual formulation of a web-based signal processing environment.
2. Development of the concept of operator encapsulation for integrated one-dimensional and two-dimensional signals.
3. Development of a Java-based framework for two-dimensional representation of one-dimensional sensor-based signals.
4. Implementation of concatenation operators for image processing applications with the ability of history profile.
5. Develop instation tools to validate Java-$MATLAB^{®}$ interaction.
6. Develop a series of algorithms in Java to implement two-dimensional cyclic operators.
7. Development of a user's guide, following usability guidelines.

Chapter 2

FUNDAMENTALS OF COMPUTATIONAL SIGNAL PROCESSING

Computational Signal Processing (CSP) deals with the formulation of computational methods in order to extract information relevant to a user. The computational methods are composed by algorithms, which are well defined procedures to solve a problem in a finite number of steps. That procedures are composed by a sequence of operations that are called operators.

Signals can be of two types: physical and mathematical. **Physical signals** are those entities which carries the information in a transmission or reception process. It can be manifested as either a matter-based, energy-based or a combination of these two components. Examples of these type of signals are any of the physical measures we can obtain from environment: temperature, pressure, humidity, etc. **Mathematical signals** are defined as numeric signals or numeric functions, it means that they have a domain and a co-domain.

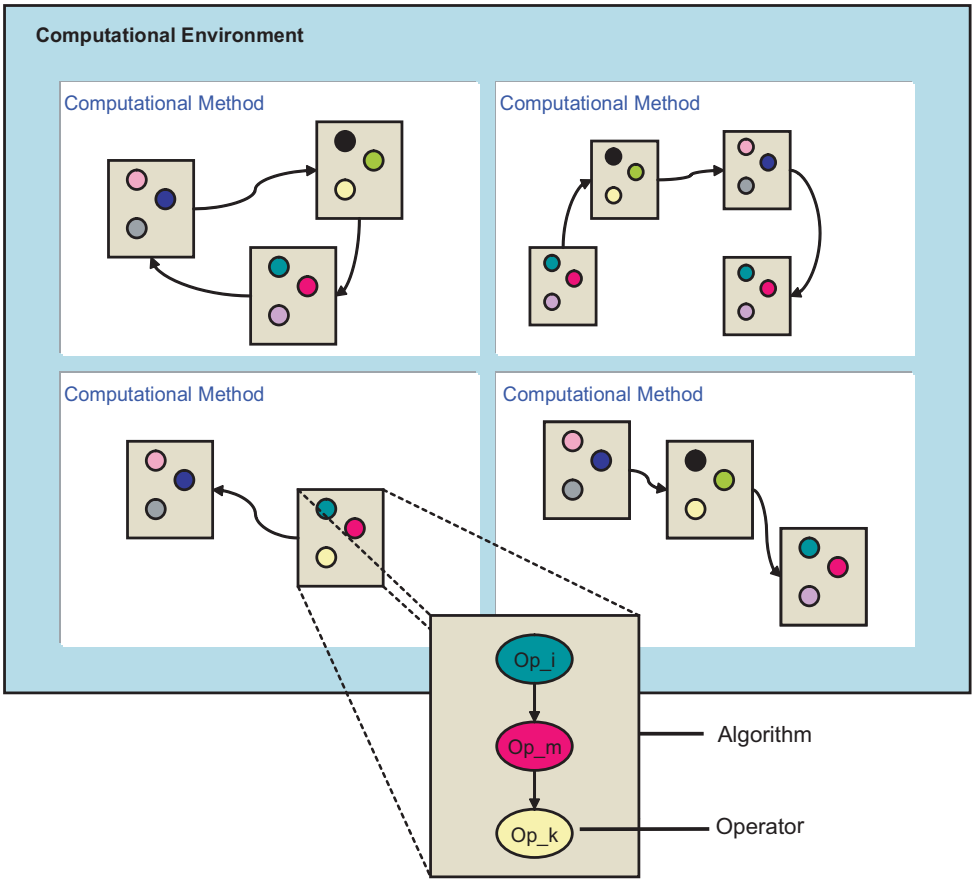

Figure 2.1: Computational Environment

It is not easy to translate physical signals to mathematical signals.

In this work digital signal processing (DSP) techniques were used to represent the signals mathematically and get the information carried by them. Mathematical basis of DSP algorithms lies on the set theory, group theory, and algebraic structures concepts which can be applied in the fields of the real numbers ($\mathbb{R}$) and the complex numbers ($\mathbb{C}$).

This chapter presents basic definitions to facilitate the understanding of these concepts. Sections 2.1 and 2.2 contain the definition of Digital Signal Processing and Multidimensional Signal Processing respectively, in order to set the mathematical representation of the signals.

Following sections are related to the basic concepts that gives the formal mathematical representation to the operations that could be done in signal processing. Most of the present definitions can be encountered in the class notes of Dr. Domingo Rodriguez [33].

2.1 Multidimensional Signal Processing

Multidimensional Signal Processing deals with the formulation of theoretical methods for the treatment of multidimensional signals in order to extract information important to a user. The treatment of the multidimensional signals is mostly of an algorithmic nature, it means, to create well-defined procedures to solve a problem in a finite number of steps.

2.1.1 Multidimensional Signal

A Multidimensional signal is any signal which admits a mathematical representation as numeric function of several independent variables. It is important to distinguish between these signals and a true physical signal, which may be conveying information about multiple observable quantities of physical entities at the same time. For example, a true physical signal may be conveying information about the temperature and pressure of a physical entity; however, this information might not be easily translated into a mathematical formulation. A numeric function of several independent variables can be denoted as: $f(x, y, z, ...)$

2.1.2 Definition of Set and Function

Before continue with the explanation of multidimensional signals, it is important to establish some definitions that will be used through this chapter, which are useful to understand the notation used to express mathematical signals and the operations that could be performed between them.

Set

In an intuitive sense, a **set** A may be defined as a collection of arbitrary objects. The collection may be finite or infinite. A set A is said to be finite if it contains a finite number of elements. Otherwise, A is said to be an infinite set. The objects contained in an arbitrary set are called the elements of the set.

For this work has been chosen the following sets of numbers:

- $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$, the set of the Integer numbers.
- $\mathbb{Z}_{\mathbb{N}} = \{0, 1, 2, ..., N-1\}$, the set of Cardinal Indexing.
- $\mathbb{R} = \{x : x \in \mathbb{R}\}$, the set of the Real numbers.
- $\mathbb{C} = \left\{z : z = x + jy; x, y \in \mathbb{R}; j = \sqrt{-1}\right\}$, the set of Complex numbers

Function

A function β is a relation with the condition that for every element $(a_k, b_n) \in \beta$, the first term of the pair is unique; that is, it appears once and only once throughout the whole set β. The word *mapping* as analogous to the word function. The notation used for a function (or mapping) is

$$f : A \longrightarrow B$$

$$a_k \longmapsto b_l = f\left(a_k\right)$$

The set A is the **domain** of the function, which is the set of the input values of f. The set B is the set of the possible output values of f and it is called the **codomain** of the function. The **range** of f is the set of all output values produced by f.

This work is based on the numeric functions that have as domain and codomain the above enumerated sets of numbers. With this in mind, it is possible make the following classification of signals.

2.1.3 Signal Classification

Real Continuous Signals of Dimension N

This signals have as domain the set of the real numbers, and as codomain also the real numbers.

$$x : \mathbb{R}^N \longrightarrow \mathbb{R}$$

$$(m_0, m_1, ..., m_{N-1}) \longmapsto x\,(m_0, m_1, ..., m_{N-1})$$

For example, consider a 2-dimensional signal. It is defined as:

$$x : \mathbb{R}^2 \longrightarrow \mathbb{R}$$

$$(m_0, m_1) \longmapsto x\,(m_0, m_1)$$

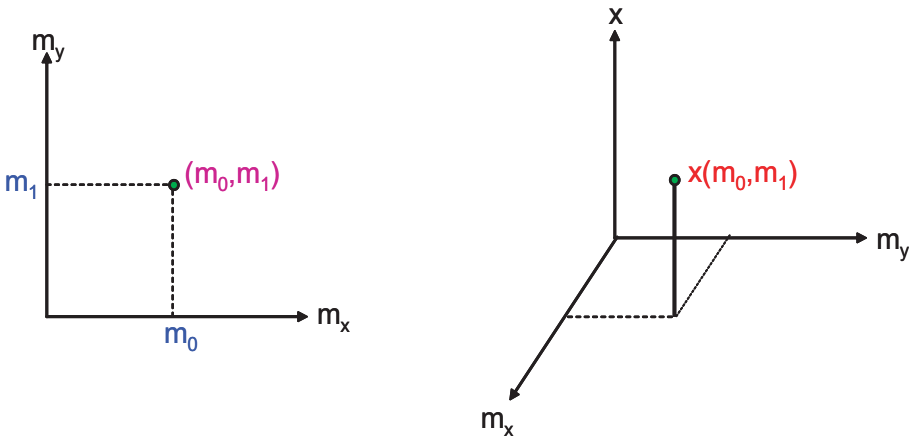

Figure 2.2: Example of a 2-Dimensional signal

If the codomain is the set of complex numbers, it is said that the signal is complex continuous of dimension N.

Real Discrete Signals of Dimension N

This signals have as domain the integer numbers, and as codomain the real numbers.

$$x : \mathbb{Z}^N \longrightarrow \mathbb{R}$$

$$(m_0, m_1, ..., m_{N-1}) \longmapsto x\,(m_0, m_1, ..., m_{N-1})$$

If the codomain is the set of complex numbers, it is said that the signal is complex discrete of dimension N.

Digital Signals

This signals have as codomain a finite set.

$$u\,(m_x, m_y) = \begin{cases} 1, & m_x \geq 1, m_y \geq 1 \\ 0, & m_x < 1, m_y < 1 \\ 0, & m_x < 1 or m_y < 1 \end{cases}$$

$$u : \mathbb{R}^{\mathbb{N}} \longrightarrow \{0, 1\}$$

$$(m_0, m_1) \longrightarrow u\,(m_0, m_1)$$

Observation. What determines if a signal is continuous or discrete is its domain, and what determines if a signal is digital, is its codomain.

Another concept that must be clear is the concept of *system*. With this concept is how has been defined the operators implemented for the CSP system.

2.1.4 Multidimensional Systems

Dudgeon [34] presents the following definition: "a ***system*** is an operator that maps one signal (the input) into another (the output)". Graphically this can be seen in the following figure:

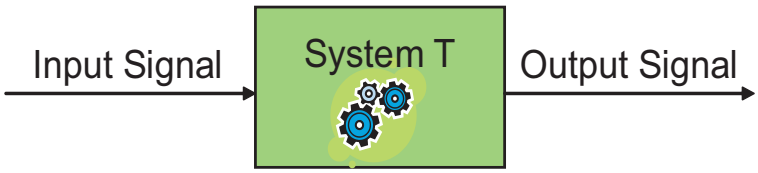

Figure 2.3: Block Diagram of a System

Digital Multidimensional Systems

These systems takes as inputs digital signals and produce as outputs digital signals as well.

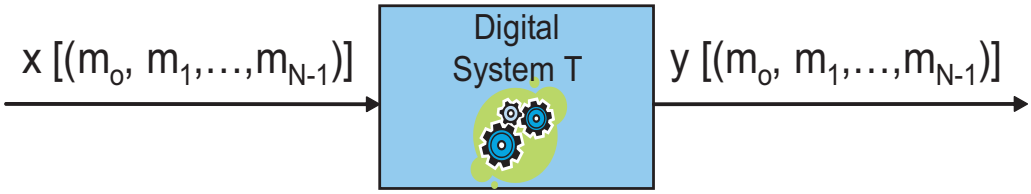

Figure 2.4: Block Diagram of a Digital System

There is another three types of systems which are very important to know, because its characteristics contribute to the implementation of operators.

Multidimensional Linear Systems

Let T be a discrete system, and let $\underline{m}= (m_0, m_1, ..., m_{N-1})$. It is said that T is *linear* if it satisfies the following conditions:

1. **Superposition**: $T\{x_1[\underline{m}] + x_2[\underline{m}]\} = T\{x_1[\underline{m}]\} + T\{x_2[\underline{m}]\}$

2. **Homogeneity**: $T\{ax_1[\underline{m}]\} = aT\{x_1[\underline{m}]\}$

This two properties can be combined to produce the following expression:

$$T\left\{ax_1\left[\underline{m}\right] + bx_2\left[\underline{m}\right]\right\} = aT\left\{x_1\left[\underline{m}\right]\right\} + bT\left\{x_2\left[\underline{m}\right]\right\}$$

Figure 2.4 illustrates by diagram blocks each side of the equality.

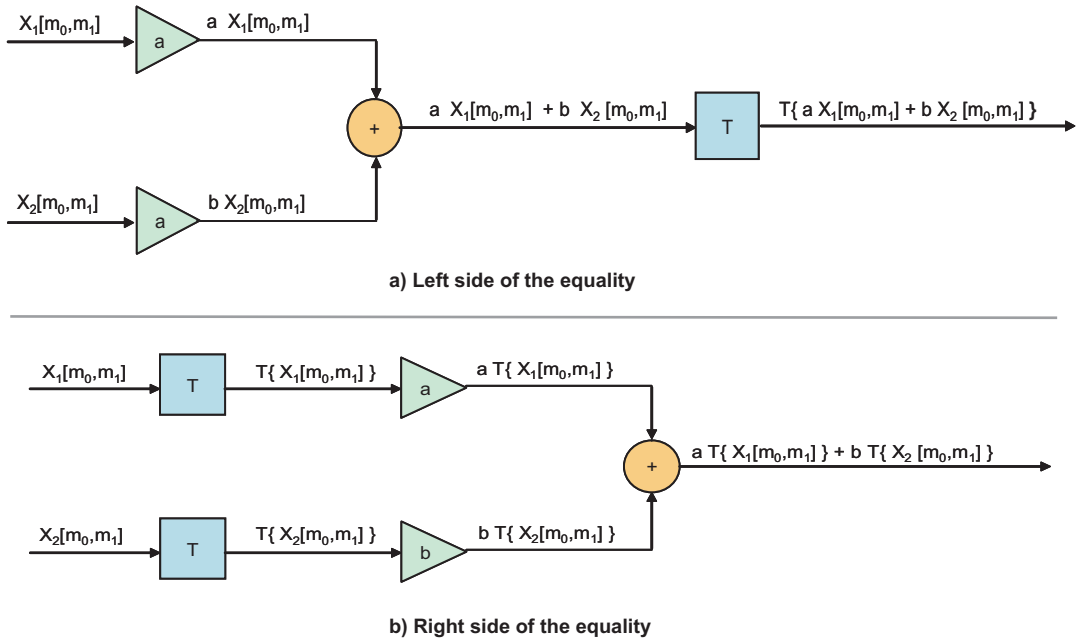

Figure 2.5: Block Diagrams for Linearity

Multidimensional Delay System

Let $\underline{m}= (m_0, m_1, ..., m_{N-1})$ and $\underline{n} = (n_0, n_1, ..., n_{N-1})$, be two elements of $\mathbb{R}^N$. The Delay system produces a dilation of $\underline{n}$ for each $\underline{m}$. Next figure illustrates this definition.

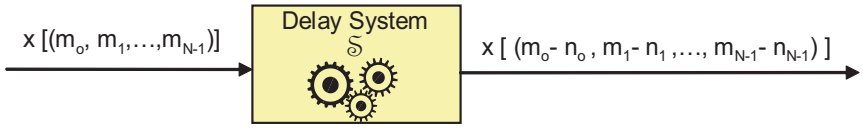

Figure 2.6: Block Diagram of a Delay System

Multidimensional Shift-Invariant Systems.

A system T is invariant if it commutes with a Delay System. Dudgeon [34] says that "a shift-invariant system is one for which a shift in the input sequence implies a corresponding shift in the output sequence".

Linear Shift-Invariant (LSI) Systems

This systems satisfy both the linearity and the shift-invariance properties described above. Mitra [35] describes this systems as "mathematically easy to analyze, and characterize, and as a consequence, easy to design. In addition, highly useful signal processing algorithms have been developed utilizing this class of systems." This systems also are called *Filters*.

Impulse Response of a LSI System

One of the most useful sequences is the ***unit sample sequence***, denoted by $\delta[\underline{m}]$ and defined by

$$\delta[\underline{m}] = \begin{cases} 1, & \underline{m} = 0 \\ 0, & \underline{m} \neq 0 \end{cases}$$

The unit sample sequence shifted by k samples is thus given by

$$\delta[\underline{m} - \underline{k}] = \begin{cases} 1, & \underline{m} = \underline{k} \\ 0, & \underline{m} \neq \underline{k} \end{cases}$$

The response of a digital filter to a unit sample sequence $\delta[\underline{m}]$ is called the **impulse response**, and it is denoted by $h[\underline{m}]$. A consequence of the linear, shift-invariant property is that an LSI system is completely specified by its impulse response, it means, that knowing the impulse response, it is possible to compute the output of the system to any arbitrary

input [35].

Using the principles of superposition, homogeneity, and shifting, it is possible to decompose any sequence into a sum of weighted and shifted unit impulses. For the particular 2-D case in [34], this can be expressed as:

$$x\left[m_0, m_1\right] = \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x\left[k_0, k_1\right] \delta\left[m_0 - k_0, m_1 - k_1\right]$$

If the sequence x is used as the input to a LSI system, called T, it will produce the output $y\left[m_0, m_1\right]$:

$$y\left[m_0, m_1\right] = T\left[\sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x\left[k_0, k_1\right] \delta\left[m_0 - k_0, m_1 - k_1\right]\right]$$

$$y\left[m_0, m_1\right] = \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x\left[k_0, k_1\right] T\left[\delta\left[m_0 - k_0, m_1 - k_1\right]\right]$$

$$y\left[m_0, m_1\right] = \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} x\left[k_0, k_1\right] h\left[m_0 - k_0, m_1 - k_1\right]$$

This last expression is known as the ***2-D convolution sum***.

Frequency Response of a LSI System

In this case it is analyzed the response of a LSI system when the input is a complex sinusoidal signal, making use also of the principles of superposition, homogeneity and shifting. Considering also the 2-D case in [34], the input signal now has this form:

$$x\left[m_0, m_1\right] = e^{\left[jw_0 m_0 + jw_1 m_1\right]}$$

the output signal can be determined by convolution in this way

$$y\left[m_0, m_1\right] = \sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} h\left[k_0, k_1\right] e^{[jw_0(m_0-k_0)+jw_1(m_1-k_1)]}$$

$$y\left[m_0, m_1\right] = e^{[jw_0m_0+jw_1m_1]} \left[\sum_{k_0=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} h\left[k_0, k_1\right] e^{[-jw_0k_0-jw_1k_1]} \right]$$

$$y\left[m_0, m_1\right] = e^{[jw_0m_0+jw_1m_1]} H\left[w_0, w_1\right]$$

The output signal is a complex sinusoid with the same frequencies as the input signal, but its amplitude and phase have been altered by the complex gain $H\left(w_0, w_1\right)$. This gain is called the system's *frequency response*, and it provides a frequency-domain description of the system. As will be explained in the next section, $H\left(w_0, w_1\right)$ corresponds to the Discrete Domain Fourier Transform of the impulse response $h\left[m_0, m_1\right]$ of the system [35].

Discrete Domain Fourier Transform: the 2-D case

Let $x\left[m_0, m_1\right]$, $m_0 \in \mathbb{Z}$, $m_1 \in \mathbb{Z}$ be a discrete signal satisfying the following condition:

$$\sum_{m_0 \in \mathbb{Z}} \sum_{m_1 \in \mathbb{Z}} x\left[m_0, m_1\right] x^*\left[m_0, m_1\right] < \infty$$

The set of all signals satisfying this condition is denoted by $l^2\left(\mathbb{Z} \times \mathbb{Z}\right)$. Thus, if $x \in l^2\left(\mathbb{Z} \times \mathbb{Z}\right)$, then it can be expressed as follows:

$$x : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{C}$$

$$[m_0, m_1] \longmapsto x\left[m_0, m_1\right]$$

The Discrete Domain Fourier Transform (DDFT) of this signal is given by the following expression:

$$X(w_0, w_1) = \sum_{m_0 \in \mathbb{Z}} \sum_{m_1 \in \mathbb{Z}} x[m_0, m_1] e^{-jw_0 m_0} e^{-jw_1 m_1}$$

The DDFT is periodic in w_0 and w_1, respectively, with periods equal to 2π in both frequency variables w_0 and w_1.

Discrete Fourier Transform: the 2-D case

The Discrete Fourier Transform (DFT) of this signal is given by the following expression:

$$X(w_{k_0}, w_{k_1}) = \sum_{m_0 \in \mathbb{Z}_{N_0}} \sum_{m_1 \in \mathbb{Z}_{N_1}} x[m_0, m_1] e^{-jw_{k_0} m_0} e^{-jw_{k_1} m_1}$$

Concepts covered in this section give a better understanding of the fundamentals of digital signal processing, especially with multidimensional signals. Before make the definitions of the operations that could be performed over that type of signals by using the systems described, is necessary to make some definitions of linear algebra and some of abstract algebra, to establish mathematical formulations for that operations.

2.2 Cartesian Products and Relations

Cartesian Product. As was seen in section 2.1, the nature of the mathematical representation of the signals is given by the domain and codomain sets in which they are defined. This helps to distinguish between continuous, discrete, and digital signals. A useful technique to show the domains of two or more variables is the **Cartesian Product**. The cartesian product of two sets A and B is a new set $A \times B$ constructed in the following manner. Every element of $A \times B$ is an ordered pair of the form (p, q) such the first term is always an element of A, or $p \in A$, and the second term is always an element of B, or $q \in B$. This can be expressed as

$$\boldsymbol{A \times B = (p, q) : p \in A, q \in B}$$

Relation. According to [36], "given a set $\boldsymbol{A}$, a ***relation*** on $\boldsymbol{A}$ can be defined as any subset of the cartesian product $\boldsymbol{A \times A}$". A relation $\boldsymbol{\rho}$ between any two sets, said $\boldsymbol{A}$ and $\boldsymbol{B}$, is a subset of its cartesian product $\boldsymbol{A \times B}$. This is indicated in the following way:

$$\boldsymbol{\rho : A \longrightarrow B}$$

$$\boldsymbol{a_k \longmapsto b_n}$$

Here the arrow $\longrightarrow$ indicates the order of the sets in the cartesian product $\boldsymbol{A \times B}$. That is, first $\boldsymbol{A}$ then $\boldsymbol{B}$. The expression $\boldsymbol{\rho : A \longrightarrow B}$ reads *the relation* $\boldsymbol{\rho}$ *such that it goes from the set* $\boldsymbol{A}$ *to the set* $\boldsymbol{B}$. The symbol $\longmapsto$ identifies the element $\boldsymbol{(a_k, b_n)}$ of the cartesian set $\boldsymbol{A \times B}$ as an element of the relation $\boldsymbol{\rho}$. The expression $\boldsymbol{a_k \longmapsto b_n}$ reads *the element* $\boldsymbol{a_k}$ *is identified with the element* $\boldsymbol{b_n}$. In this case, $\boldsymbol{a_k \in A}$ is related to $\boldsymbol{b_n \in B}$.

2.3 Binary Operations and Multiplicative Sets

Having in mind the concepts of set, function, cartesian product and relation, it is possible to define the concept of binary operation, which is very important to continue with the presentation of the algorithms of the operations established for multidimensional signal processing.

Binary Operation. Let $\boldsymbol{A}$ be a finite set. $\boldsymbol{\beta}$ is a binary operation defined on $\boldsymbol{A}$ if $\boldsymbol{\beta}$ is

a mapping or function of the form

$$\boldsymbol{\beta : A \times A \longrightarrow A}$$

$$\boldsymbol{(a_k, al) \longmapsto \beta\,(a_k, al) \equiv a_m}$$

In this definition $\boldsymbol{a_k, a_l, a_m}$ are any arbitrary elements of A. Usually, $\boldsymbol{a_k}$ and $\boldsymbol{a_l}$ are called the left operand of the binary operation and the right operand of the binary operation, respectively.

Multiplicative Set. A finite set $\boldsymbol{A}$ is called a multiplicative set if is possible identify a binary operation on this set. The operation itself need not be a multiplication operation.

2.4 Basic Algebraic Structures

Naylor [36] defines **Algebraic System** as the set of binary operations for any set. For example, the set $\mathbb{N}$ and the sum operation $+$. The structure of the system is instead characterized by a set of axioms, and all systems satisfying these axioms are grouped together in a set called an **Algebraic Structure**.

"Algebraic structures and number theory underlie much of the development of modern techniques of digital signal processing (DSP), for this is important study DSP with a formal survey of these mathematical fundamentals" [37].

Following subsections present some of the most used algebraic structures, starting from the more simple to the most complex.

2.4.1 Finite Semigroup, Finite Group, Finite Subgroup, Abelian Group

Finite Semigroup. The set $\boldsymbol{A}$ is a finite semigroup if $\boldsymbol{A}$ is a multiplicative set with an associative binary operation. That is, if $\boldsymbol{\beta}$ is the binary operation, then

$$\boldsymbol{\beta : A \times A \longrightarrow A}$$

$$\boldsymbol{(a_k, al) \longmapsto \beta\,(a_k, al) \equiv a_m}$$

Finite Group. A set $\boldsymbol{A}$ is a *finite group* if it is a semigroup with the following additional properties:

1. For every element $\boldsymbol{a_k \in A}$, always find another element $\boldsymbol{a_n \in A}$ such that

$$\boldsymbol{a_k * a_n = e}$$

2. There exists a unique element $\boldsymbol{e \in A}$ such that for every element $\boldsymbol{a_l \in A}$, we have

$$\boldsymbol{e * a_l = a_l * e = e}$$

The element $\boldsymbol{e \in A}$ is the identity element of the group $\boldsymbol{A}$. If $\boldsymbol{a_k * a_n = e}$, $\boldsymbol{a_n}$ is the inverse of $\boldsymbol{a_k}$, denoted by $\boldsymbol{a_k^{-1}}$, and write $\boldsymbol{a_k^{-1} = a_n}$.

Finite Subgroup. Let $\boldsymbol{G}$ be a finite group with binary operation $*$. Let $\boldsymbol{H}$ be a subset of $\boldsymbol{G}$. $\boldsymbol{H}$ is a subgroup of $\boldsymbol{G}$ if for any two elements $\boldsymbol{h_k, h_l \in H}$, have $\boldsymbol{h_k * h_l = h_m \in H}$; and $\boldsymbol{h_r \in H}$ implies $\boldsymbol{h_r^{-1} \in H}$.

Abelian Group. A group $\boldsymbol{A}$ is an abelian group if the binary operation of the group is

a commutative operation; if $\boldsymbol{a_k, a_l} \in \boldsymbol{A}$ are any two elements of $\boldsymbol{A}$, then

$$\boldsymbol{a_k a_l = a_l a_k}$$

is always satisfied. The group is called an *additive abelian group* if the internal binary operation of the group is the additive operation.

Homomorphism. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be any two arbitrary groups with binary operations $\oplus$ and $\odot$, respectively. A *homomorphism* from the group $\boldsymbol{A}$ into $\boldsymbol{B}$ is a mapping $\boldsymbol{\mu : A \longmapsto B}$ such that if

$$\boldsymbol{a_k, a_l \in A, \mu\left(a_k\right), \mu\left(a_l\right) \in B}$$

then

$$\boldsymbol{\mu\left(a_k \oplus a_l\right) = \mu\left(a_k\right) \odot \mu\left(a_l\right)}$$

The homomorphism $\boldsymbol{\mu}$ is called an *epimorphism* if it is onto; it is called *monomorphism* if it is one-to-one; and it is called an *isomorphism* if it is both one-to-one and onto. If $\boldsymbol{A = B}$ then $\boldsymbol{\mu : A \longrightarrow A}$ is called an *endomorphism*. If $\boldsymbol{\mu : A \longrightarrow A}$ is an isomorphism, then it is called an *automorphism*.

2.4.2 Rings, Fields

Ring. A set $\boldsymbol{A}$ is a ring if it is an additive abelian group with a binary multiplication operation defined. For every ordered pair of elements $\boldsymbol{\left(a_k, a_l\right), a_k, a_l \in A}$, a multiplication operation results in an element $\boldsymbol{a_k a_l \in A}$. This multiplication operation has the following properties. It is a commutative operation, and associative as well. It is also distributive over the addition operation. A multiplicative identity, written as element $\mathbf{1}$, exists in $\boldsymbol{A}$ such that for any element $\boldsymbol{a_k \in A, 1 a_k = a_k 1 = a_k}$.

Field. A set $\boldsymbol{A}$ is called a field if we can identify two internal binary operations which we term addition and multiplication. The set $\boldsymbol{A}$ behaves as an additive abelian group under the addition operation. The set $\boldsymbol{A}^*$ behaves as an abelian group under the multiplication operation. In this work, it has been used the field of the real numbers ($\mathbb{R}$) and the set of the complex numbers ($\mathbb{C}$).

2.4.3 Vector Spaces

.

Greub [38] defines a ***Vector (Linear) Space***, $\boldsymbol{V}$, over the field $\boldsymbol{\Gamma}$ as a set of elements $\{\boldsymbol{v_k}, \boldsymbol{v_l}, \ldots\}$ called *vectors* with the following algebraic structure:

1. $\boldsymbol{V}$ is an additive group; that is, there is a fixed mapping

$$+ : \boldsymbol{V} \times \boldsymbol{V} \longrightarrow \boldsymbol{V}$$

$$(\boldsymbol{v_k}, \boldsymbol{v_l}) \longmapsto + (\boldsymbol{v_k}, \boldsymbol{v_l}) = (\boldsymbol{v_k} + \boldsymbol{v_l}) = \boldsymbol{v_m}$$

   and satisfying the following axioms:

   (a) $(\boldsymbol{v_k} + \boldsymbol{v_l}) + \boldsymbol{v_r} = \boldsymbol{v_k} + (\boldsymbol{v_l} + \boldsymbol{v_r})$ (*Associative Law*)

   (b) $(\boldsymbol{v_k} + \boldsymbol{v_l}) = (\boldsymbol{v_l} + \boldsymbol{v_k})$ (*Commutative Law*)

   (c) there exists a zero-vector $\mathbf{0}$; i.e., a vector such that $\boldsymbol{x} + \mathbf{0} = \mathbf{0} + \boldsymbol{v_k} = \boldsymbol{v_k}$ for every $\boldsymbol{v_k} \in \boldsymbol{V}$.

   (d) To every vector $\boldsymbol{v_k}$ there is a vector $-\boldsymbol{v_k}$ such that $\boldsymbol{v_k} + (-\boldsymbol{v_k}) = \mathbf{0}$.

2. There is a fixed mapping

$$\bullet : \boldsymbol{V} \times \boldsymbol{V} \longrightarrow \boldsymbol{V}$$

$$(\lambda, v_k) \longmapsto \bullet\,(\lambda, v_k) = \lambda v_k = v_s$$

and satisfying the axioms:

(a) $(\lambda\mu)\, v_k = \lambda\,(\mu v_k)$ (*Associative Law*)

(b) $(\lambda + \mu)\, v_k = \lambda v_k + \mu v_k$

$\lambda\,(x + y) = \lambda x + \lambda y$ (*Distributive Laws*)

(c) $1 \cdot v_k = v_k$ (1 is the unit element of Γ)

Linear Combinations. A linear combination of $\{v_k, v_l, \ldots, v_n\}$ is an expression of the form

$$c_k v_k + c_l v_l + \ldots + c_n v_n$$

where the c's are scalars $\in \Gamma$.

A subset $S \subset V$ is called a ***system of generators*** for V if every vector $v \in V$ is a linear combination of vectors of S.

Span Set. Let $\{v_k, v_l, \ldots, v_n\}$ be vectors in the space V. The ***span*** of these vectors, denoted by $span v_k, v_l, \ldots, v_n$, is the subset of V consisting of all possible linear combinations of these vectors:

$$span\,\{v_k, v_l, \ldots, v_n\} = \{c_k v_k, c_l v_l, \ldots, c_n v_n\}$$

where $c_k, c_l, \ldots, c_n$ are scalars $\in \Gamma$.

Linear Independence. A set of vectors $V = \{v_k, v_l, \ldots, v_n\}$ is said to be a linearly independent set whenever the only solution for the scalars α_i in the homogeneous equation

$$\alpha_k v_k + \alpha_l v_l + \ldots + \alpha_n v_n = 0$$

is the trivial solution $\alpha_k = \alpha_l = \ldots = \alpha_n = 0$

Basis. A basis for the vector space V is a spanning set of vectors $\{v_k, v_l, \ldots, v_n\}$ which is linearly independent. The standard basis of $\mathbb{R}^N$ or $\mathbb{C}^N$ is the set $\{e_0, e_1, \ldots, e_{N-1}\}$ where e_j is the column vector of size N whose jth entry is 1 and all other entries is 0.

$$e_0 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; e_1 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}; \ldots e_N = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Standard vector spaces alway have a basis. Given a vector space V, not always can be sure that a basis for it exists.

Linear Operator. A function that maps elements of one vector space into another, say $f : V \longrightarrow W$ is sometimes called an ***operator***, a ***system***, or ***transformation***.

2.4.4 Linear Algebra

A linear algebra over the field Γ is a vector space V denoted as $(V, +, \bullet)$ with an additional operation called a vector multiplication or signal multiplication operation defined as follows:

$$*V \times V \longrightarrow V$$

$$(v_k, v_l) \longmapsto *(v_k, v_l) = (v_k * v_l) = v_m$$

This signal multiplication operation has the following three properties:

1. **Associative Property**. For $\boldsymbol{v_k, v_l, v_r \in V}$ have

$$\boldsymbol{(v_k * v_l) * v_r = v_k * (v_l * v_r)}$$

2. **Distributive Property with respect to the vector addition**. For $\boldsymbol{v_k, v_l, v_r \in V}$ have

$$\boldsymbol{v_k * (v_l + v_r) = v_k * v_l + v_k * v_r}$$

3. **Distributive Property with respect to the scalars**. For $\boldsymbol{v_k, v_l \in V}$ and $\boldsymbol{a \in \Gamma}$ have

$$\boldsymbol{a\,(v_k + v_l) = (av_k) * v_l = v_k * (av_l)}$$

2.5 Space of Signals

We term a set $\boldsymbol{A}$ a signal space, or simply a space, if each element of $\boldsymbol{A}$ is a signal or function.

Space of Finite Complex Signals $\boldsymbol{l\,(\mathbb{Z}_N)}$: 1-D case. Let $\boldsymbol{l\,(\mathbb{Z}_N)}$ be the set of all complex signals of length $\boldsymbol{N}$ defined as follows:

$$\boldsymbol{x : \mathbb{Z}_N \longrightarrow \mathbb{C}}$$

$$\boldsymbol{n \longmapsto x\,[n]}$$

The Standard Basis Set for $\boldsymbol{l\,(\mathbb{Z}_N)}$: 1-D case. Let $\boldsymbol{\Delta_N \in l\,(\mathbb{Z}_N)}$ be the set of $\boldsymbol{N}$ complex signals called the *standard basis set*. We proceed to describe this set:

$$\boldsymbol{\Delta_N = \left\{\delta_{\{k\}} : \delta_{\{k\}} = 1, n = k; \delta_{\{k\}} = 0, n \neq k; n, k \in \mathbb{Z}_N\right\}}$$

$$\boldsymbol{\Delta_N = \left\{\delta_{\{0\}}, \delta_{\{1\}}, \delta_{\{2\}}, \ldots, \delta_{\{N-1\}}\right\}}$$

$$\boldsymbol{\delta_{\{m\}}[n], n \in \mathbb{Z}_N}$$

$$\boldsymbol{\delta_{\{m\}}[n] = 1}$$

For example, having $\boldsymbol{N = 4}$

$$\boldsymbol{\Delta_4 = \left\{\delta_{\{0\}}, \delta_{\{1\}}, \delta_{\{2\}}, \delta_{\{3\}}\right\}}$$

$$\boldsymbol{\delta_{\{0\}} = \left\{\delta_{\{0\}}[0], \delta_{\{0\}}[1], \delta_{\{0\}}[2], \delta_{\{0\}}[3]\right\}}$$

$$\boldsymbol{\delta_{\{1\}} = \left\{\delta_{\{1\}}[0], \delta_{\{1\}}[1], \delta_{\{1\}}[2], \delta_{\{1\}}[3]\right\}}$$

$$\boldsymbol{\delta_{\{2\}} = \left\{\delta_{\{2\}}[0], \delta_{\{2\}}[1], \delta_{\{2\}}[2], \delta_{\{2\}}[3]\right\}}$$

$$\boldsymbol{\delta_{\{3\}} = \left\{\delta_{\{3\}}[0], \delta_{\{3\}}[1], \delta_{\{3\}}[2], \delta_{\{3\}}[3]\right\}}$$

We represent signals in $\boldsymbol{l(\mathbb{Z}_N)}$ as column vectors of length $\boldsymbol{N}$ and for this reason call the elements of the space $\boldsymbol{l(\mathbb{Z}_N)}$ vectors of the space.

$$\boldsymbol{\delta_{\{2\}} = \begin{bmatrix} \delta_{\{2\}}[0] \\ \delta_{\{2\}}[1] \\ \delta_{\{2\}}[2] \\ \delta_{\{2\}}[3] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}}$$

Space of Finite Complex Signals $\boldsymbol{l^2(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1})}$: 2-D case. Let $\boldsymbol{l^2(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1})}$ be the set of all complex signals of length $\boldsymbol{N_0 \times N_1}$ defined as follows:

$$\boldsymbol{x} : \boldsymbol{l^2}\left(\mathbb{Z}_{\boldsymbol{N_0}} \times \mathbb{Z}_{\boldsymbol{N_1}}\right) \longrightarrow \mathbb{C}$$

$$\left(\boldsymbol{n_0}, \boldsymbol{n_1}\right) \longmapsto \boldsymbol{x}\left[\boldsymbol{n_0}, \boldsymbol{n_1}\right]$$

The Standard Basis Set for $l^2\left(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1}\right)$: 2-D case. Let $\boldsymbol{\Delta}_{\boldsymbol{N_0} \times \boldsymbol{N_1}} \in \boldsymbol{l^2}\left(\mathbb{Z}_{\boldsymbol{N_0}} \times \mathbb{Z}_{\boldsymbol{N_1}}\right)$ be the set of $\boldsymbol{N_0} \times \boldsymbol{N_1}$ complex signals called the *standard basis set*. We proceed to describe this set:

$$\boldsymbol{\Delta}_{\boldsymbol{N_0} \times \boldsymbol{N_1}} = \left\{\boldsymbol{\delta}_{\{\boldsymbol{k_0}, \boldsymbol{k_1}\}} : \boldsymbol{\delta}_{\{\boldsymbol{k_0}, \boldsymbol{k_1}\}} = \boldsymbol{1}, \boldsymbol{n_0} = \boldsymbol{k_0}, \boldsymbol{n_1} = \boldsymbol{k_1}; \boldsymbol{\delta}_{\{\boldsymbol{k_0}, \boldsymbol{k_1}\}} = \boldsymbol{0}, \boldsymbol{n_0} \neq \boldsymbol{k_0}, \boldsymbol{n_1} \neq \boldsymbol{k_1}\right\}$$

2.5.1 Binary Operations defined over $l\left(\mathbb{Z}_N\right)$: 1-D case

We can define many binary operations over the space $\boldsymbol{l}\left(\mathbb{Z}_{\boldsymbol{N}}\right)$ of particular importance are the operations termed *Cyclic Convolution* and *Hadamard Product*.

- **Cyclic Convolution Binary Operation over $l\left(\mathbb{Z}_N\right)$**. This operation is defined as follows:

$$\odot_{\boldsymbol{N}} \boldsymbol{l}\left(\mathbb{Z}_{\boldsymbol{N}}\right) \times \boldsymbol{l}\left(\mathbb{Z}_{\boldsymbol{N}}\right) \longrightarrow \boldsymbol{l}\left(\mathbb{Z}_{\boldsymbol{N}}\right)$$

$$\left(\boldsymbol{x}, \boldsymbol{h}\right) \longmapsto \boldsymbol{y} = \odot_{\boldsymbol{N}}\left(\boldsymbol{x}, \boldsymbol{h}\right)$$

  where:

$$\boldsymbol{y}\left[\boldsymbol{n}\right] = \left(\odot_{\boldsymbol{N}}\left(\boldsymbol{x}, \boldsymbol{h}\right)\right)\left[\boldsymbol{n}\right] = \sum_{\boldsymbol{k} \in \mathbb{Z}_{\boldsymbol{N}}} \boldsymbol{x}\left[\boldsymbol{k}\right] \boldsymbol{h}\left[\left\langle \boldsymbol{n} - \boldsymbol{k} \right\rangle_{\boldsymbol{N}}\right]$$

$$= \sum_{\boldsymbol{k} \in \mathbb{Z}_{\boldsymbol{N}}} \boldsymbol{h}\left[\boldsymbol{k}\right] \boldsymbol{x}\left[\left\langle \boldsymbol{n} - \boldsymbol{k} \right\rangle_{\boldsymbol{N}}\right]$$

  Here, $\left\langle \boldsymbol{\rho} \right\rangle_{\boldsymbol{N}}$ denotes a module arithmetic operation; that is, $\left\langle \boldsymbol{\rho} \right\rangle_{\boldsymbol{N}} \equiv$ remainder $\left(\frac{\boldsymbol{p}}{\boldsymbol{N}}\right)$

Normally, we use the notation $\boldsymbol{x} \odot_{\boldsymbol{N}} \boldsymbol{h}$ instead of $\odot_{\boldsymbol{N}} (\boldsymbol{x}, \boldsymbol{h})$.

Also, we know that $\odot_{\boldsymbol{N}} \subset (\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$

- **Hadamard Product Binary Operation over $\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$.** This operation is defined as follows:

$$\odot_{\boldsymbol{N}} \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \longrightarrow \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$$

$$(\boldsymbol{x}, \boldsymbol{v}) \longmapsto \boldsymbol{s} = \odot_{\boldsymbol{N}} (\boldsymbol{x}, \boldsymbol{v})$$

where:

$$\boldsymbol{y}[\boldsymbol{n}] = (\odot_{\boldsymbol{N}} (\boldsymbol{x}, \boldsymbol{h}))[\boldsymbol{n}] = \boldsymbol{x}[\boldsymbol{n}] \bullet \boldsymbol{v}[\boldsymbol{n}], \boldsymbol{n} \in \mathbb{Z}_{\boldsymbol{N}}$$

Normally, we use the notation $\boldsymbol{x} \odot_{\boldsymbol{N}} \boldsymbol{v}$ instead of $\odot_{\boldsymbol{N}} (\boldsymbol{x}, \boldsymbol{v})$.

Also, we know that $\odot_{\boldsymbol{N}} \subset (\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$

- **Shift Operator over $\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$.** The cyclic Shift Operator $\boldsymbol{S}_{\boldsymbol{N}}$ over the space $\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$ with respect to the standard basis $\boldsymbol{\Delta}_{\boldsymbol{N}}$ is defined as follows:

$$\boldsymbol{S}_{\boldsymbol{N}} : \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \longrightarrow \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$$

$$\boldsymbol{\delta}_{\boldsymbol{k}} \longmapsto \boldsymbol{S}_{\boldsymbol{N}} \{\boldsymbol{\delta}_{\boldsymbol{k}}\} = \boldsymbol{\delta}_{\langle \boldsymbol{k}+1 \rangle_{\boldsymbol{N}}}$$

The matrix representation of this operator with respect to the standard basis is given by the following expression:

$$\boldsymbol{S}_{\boldsymbol{N}} = \left[\mathscr{S}_{\boldsymbol{N}} \{\boldsymbol{\delta}_{\{0\}}\} \ \mathscr{S}_{\boldsymbol{N}} \{\boldsymbol{\delta}_{\{1\}}\} \ \mathscr{S}_{\boldsymbol{N}} \{\boldsymbol{\delta}_{\{2\}}\} \ldots \mathscr{S}_{\boldsymbol{N}} \{\boldsymbol{\delta}_{\{N-1\}}\}\right]$$

- **Reflection Operator or Cyclic Index Reversal Operator over $\boldsymbol{l}(\mathbb{Z}_N)$.** The cyclic reflection operator or index reversal operator over the space $\boldsymbol{l}(\mathbb{Z}_N)$ is defined as follows:

$$\mathscr{R}_N : \boldsymbol{l}(\mathbb{Z}_N) \longrightarrow \boldsymbol{l}(\mathbb{Z}_N)$$

$$\boldsymbol{x} \longmapsto \mathscr{R}_N\{\boldsymbol{x}\}$$

where:

$$(\mathscr{R}_N\{\boldsymbol{x}\})[\boldsymbol{n}] = \boldsymbol{x}^{(-)}[\boldsymbol{n}] = \boldsymbol{x}[\langle -\boldsymbol{n}\rangle_N]$$

2.5.2 Binary Operations defined over $\boldsymbol{l^2}(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1})$: 2-D case

- **Cyclic Convolution Binary Operation over $\boldsymbol{l^2}(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1})$.** This operation is defined as follows:

$$\odot_{N_0 \times N_1} : \boldsymbol{l^2}(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1}) \times \boldsymbol{l^2}(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1}) \longrightarrow \boldsymbol{l^2}(\mathbb{Z}_{N_0} \times \mathbb{Z}_{N_1})$$

$$(\boldsymbol{x}, \boldsymbol{h}) \longmapsto \boldsymbol{y} = \odot_{N_0 \times N_1}(\boldsymbol{x}, \boldsymbol{h})$$

where:

$$\boldsymbol{y}[\boldsymbol{n_0}, \boldsymbol{n_1}] = (\odot_{N_0 \times N_1}(\boldsymbol{x}, \boldsymbol{h}))[\boldsymbol{n_0}, \boldsymbol{n_1}]$$

$$\boldsymbol{y}[\boldsymbol{n_0}, \boldsymbol{n_1}] = \sum_{\boldsymbol{k_1} \in \mathbb{Z}_{N_1}} \sum_{\boldsymbol{k_0} \in \mathbb{Z}_{N_0}} \boldsymbol{x}[\boldsymbol{k_0}, \boldsymbol{k_1}]\, \boldsymbol{h}\left[\langle \boldsymbol{n_0} - \boldsymbol{k_0}\rangle_{N_0}, \langle \boldsymbol{n_1} - \boldsymbol{k_1}\rangle_{N_1}\right]$$

$$\boldsymbol{y}[\boldsymbol{n_0}, \boldsymbol{n_1}] = \sum_{\boldsymbol{k_1} \in \mathbb{Z}_{N_1}} \sum_{\boldsymbol{k_0} \in \mathbb{Z}_{N_0}} \boldsymbol{x}[\boldsymbol{k_0}, \boldsymbol{k_1}]\, \boldsymbol{h}\left[\langle \boldsymbol{n_0} - \boldsymbol{k_0}\rangle_{N_0}, \langle \boldsymbol{n_1} - \boldsymbol{k_1}\rangle_{N_1}\right]$$

Here, $\langle \boldsymbol{\rho} \rangle_{\boldsymbol{N}}$ denotes a module arithmetic operation; that is, $\langle \boldsymbol{\rho} \rangle_{\boldsymbol{N}} \equiv$ remainder $\left(\frac{\boldsymbol{p}}{\boldsymbol{N}}\right)$

Normally, we use the notation $\boldsymbol{x} \odot_{\boldsymbol{N}} \boldsymbol{h}$ instead of $\odot_{\boldsymbol{N}} (\boldsymbol{x}, \boldsymbol{h})$.

Also, we know that $\odot_{\boldsymbol{N}} \subset (\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$

- **Hadamard Product Binary Operation over $\boldsymbol{l^2}(\mathbb{Z}_{\boldsymbol{N_0}} \times \mathbb{Z}_{\boldsymbol{N_1}})$.** This operation is defined as follows:

$$\odot_{\boldsymbol{N_0} \times \boldsymbol{N_1}} \boldsymbol{l^2}(\mathbb{Z}_{\boldsymbol{N_0}} \times \mathbb{Z}_{\boldsymbol{N_1}}) \times \boldsymbol{l^2}(\mathbb{Z}_{\boldsymbol{N_0}} \times \mathbb{Z}_{\boldsymbol{N_1}}) \longrightarrow \boldsymbol{l^2}(\mathbb{Z}_{\boldsymbol{N_0}} \times \mathbb{Z}_{\boldsymbol{N_1}})$$

$$(\boldsymbol{x}, \boldsymbol{v}) \longmapsto \boldsymbol{s} = \odot_{\boldsymbol{N_0} \times \boldsymbol{N_1}} (\boldsymbol{x}, \boldsymbol{v})$$

where:

$$\boldsymbol{y}[\boldsymbol{n_0}, \boldsymbol{n_1}] = (\odot_{\boldsymbol{N_0} \times \boldsymbol{N_1}} (\boldsymbol{x}, \boldsymbol{h}))[\boldsymbol{n_0}, \boldsymbol{n_1}] = \boldsymbol{x}[\boldsymbol{n_0}, \boldsymbol{n_1}] \bullet \boldsymbol{v}[\boldsymbol{n_0}, \boldsymbol{n_1}], \boldsymbol{n_0} \in \mathbb{Z}_{\boldsymbol{N_0}}, \boldsymbol{n_1} \in \mathbb{Z}_{\boldsymbol{N_1}}$$

Normally, we use the notation $\boldsymbol{x} \odot_{\boldsymbol{N}} \boldsymbol{v}$ instead of $\odot_{\boldsymbol{N}} (\boldsymbol{x}, \boldsymbol{v})$.

Also, we know that $\odot_{\boldsymbol{N}} \subset (\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})) \times \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$

- **Shift Operator over $\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$.** The cyclic Shift Operator $\boldsymbol{S_N}$ over the space $\boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$ with respect to the standard basis $\boldsymbol{\Delta_N}$ is defined as follows:

$$\boldsymbol{S_N} : \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}}) \longrightarrow \boldsymbol{l}(\mathbb{Z}_{\boldsymbol{N}})$$

$$\boldsymbol{\delta_k} \longmapsto \boldsymbol{S_N}\{\boldsymbol{\delta_k}\} = \boldsymbol{\delta}_{\langle \boldsymbol{k}+1 \rangle_{\boldsymbol{N}}}$$

The matrix representation of this operator with respect to the standard basis is given

by the following expression:

2.6 Inner Product Spaces

An ***inner product*** in a real vector space $\boldsymbol{E}$ is a bilinear function $\boldsymbol{(,)}$ having the following properties [38]:

1. Symmetry: $\boldsymbol{(x, y) = (y, x)}$

2. Positive definiteness: $\boldsymbol{(x, x) \geq 0}$, and $\boldsymbol{(x, x) = 0}$ only for the vector $\boldsymbol{x = 0}$.

A vector space in which an inner product is defined is called an *inner product space*. An inner product space of finite dimension is also called a *Euclidean Space*.

The norm $\boldsymbol{|x|}$ of a vector $\boldsymbol{x \in E}$ is defined as the positive square-root

$$\boldsymbol{|x| = \sqrt{(x, x)}}$$

A *unit vector* is a vector with the norm $\mathbf{1}$. The set of all unit vectors is called the *unit-sphere*.

Orthogonality. Two vectors $\boldsymbol{x \in E}$ and $\boldsymbol{y \in E}$ are said to be *orthogonal* if $\boldsymbol{(x, y) = 0}$. A system of $\boldsymbol{p}$ vectors $\boldsymbol{x_v \neq 0}$ in which any two vectors $\boldsymbol{x_v}$ and $\boldsymbol{x_u (v \neq u)}$ are orthogonal, is linearly independent.

2.7 Hilbert Spaces

Let $\boldsymbol{L^2}(\mathbb{C}))$ be the space of all energy complex signals of the form:

$$\boldsymbol{x : \mathbb{R} \longrightarrow \mathbb{C}}$$

$$t \longmapsto x(t)$$

This space has an inner product defined as follows:

$$\langle x, y \rangle = \int_{\infty}^{\infty} x(t)\, y^*(t)\, dt \neq \langle y, x \rangle$$

All signals in this space must satisfy the following condition:

$$\langle x, x \rangle = \int_{\infty}^{\infty} x(t)\, x^*(t)\, dt < \infty$$

The linear space with the inner product defined as described above form a space called a Hilbert Space of complex continuous functions with finite energy.

Chapter 3

DIGITAL IMAGE PROCESSING

Interest in Digital Image Processing (DIP) methods stems from two principal application areas: improvement of pictorial information for human interpretation, and processing of image data for storage, transmission, and representation for autonomous machine interception [39].

DIP refers to processing digital images by means of a digital computer. A digital image may be defined as a 2-D function $\boldsymbol{f(x,y)}$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are spatial (plane) coordinates finite and discrete, as well as the amplitude of $\boldsymbol{f}$ at any pair of coordinates $\boldsymbol{(x,y)}$, called the intensity or gray level of the image at that point.

A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, and pixels.

Fundamentals steps in DIP

Gonzalez [40] identify two broad categories of methods to treat images:

1. Methods whose input and output are images.

    - **Image Acquisition**. The origin of digital images.
    - **Image Enhancement**. To highlight certain features of interest in an image.
    - **Image Restoration**. It uses restoration techniques that tend to be based on mathematical or probabilistic models of image degradation.
    - **Morphological Processing**. Deals with tools for extracting image components that are useful in the representation and description of shape.

2. Methods whose inputs may be images, but whose outputs are attribute extracted from those images.

    - **Segmentation**. Procedures partition an image into the constituent parts or objects.
    - **Representation and Description**. It deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.
    - **Recognition**. It is the process that assigns a label to an object on its descriptors.

The number of gray levels typically is an integer power of 2: $\boldsymbol{L = 2^k}$, where $\boldsymbol{k}$ is the number of bits of quantization. When an image can have $\mathbf{2^k}$ gray levels, it is common to refer to the image as a "k-bit image". For example, an image with 256 possible gray-level values is called an 8-bit image.

3.1 Image Representation

The information contained in images can be represented in different ways, being the spatial and the frequency representation the most known. Spatial representation refers to the rep-

resentation of images as two dimensional (2D) arrays, in which each element of the matrix is called a **pixel**. For this representation is used the common notation for matrices.

Figure 3.1 shows an example of a spatial representation of a very simple image of an unitary impulse. Black pixels are represented by zeros and the white pixels by ones. In the lower left side appears the 3D representation of the unitary impulse.

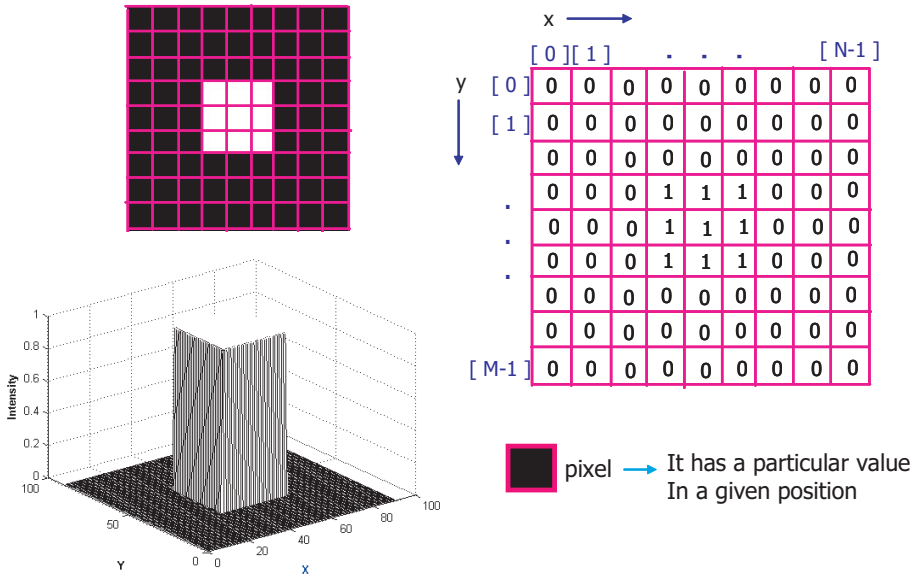

Figure 3.1: Spatial Representation of Images

If the image contains $\boldsymbol{M} \times \boldsymbol{N}$ pixels, it is represented by an $\boldsymbol{M} \times \boldsymbol{N}$ matrix, the index $\boldsymbol{n_x}$ runs from $\mathbf{0}$ to $\boldsymbol{N-1}$, and index $\boldsymbol{n_y}$ from $\mathbf{0}$ to $\boldsymbol{M-1}$. Following is the general matrix representation of a digital image:

$$x[n_x, n_y] = \begin{Bmatrix} x[0,0] & x[0,1] & x[0,2] & \dots & x[0,M-1] \\ x[1,0] & x[1,1] & x[1,2] & \dots & x[1,M-1] \\ x[2,0] & x[2,1] & x[2,2] & \dots & x[2,M-1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x[N-1,0] & x[N-1,1] & x[N-1,2] & \dots & x[N-1,M-1] \end{Bmatrix}$$

The representation in the frequency domain of a digital image is obtained by applying the Discrete Fourier Transform (DFT) to the matrix representing the image, using the mathematical expression defined in section 2.1.

3.2 Signed Representation of Images

In this research work was used grayscale images. In this type of images intensities of pixels are shades of gray ranging from black to white. This can be viewed clearly in figure 3.2.

One of the problems confronted is to find the appropriated range of gray values to represent graphically the output from the signal operators, especially if that outputs are negative numbers. Jahne [39] gives an approach to solve this problem in the following way. Each pixel in images of 256 gray values occupies 8 bits, which causes that negative pixels will take large positive values. "In a 8-bit representation, is possible to convert unsigned numbers into signed numbers by subtracting **128**:

$$q' = (q - 128)\, mod 256; 0 \leq q' \leq 256$$

Essentially, gray values are regarded in this representation as a deviation from a mean

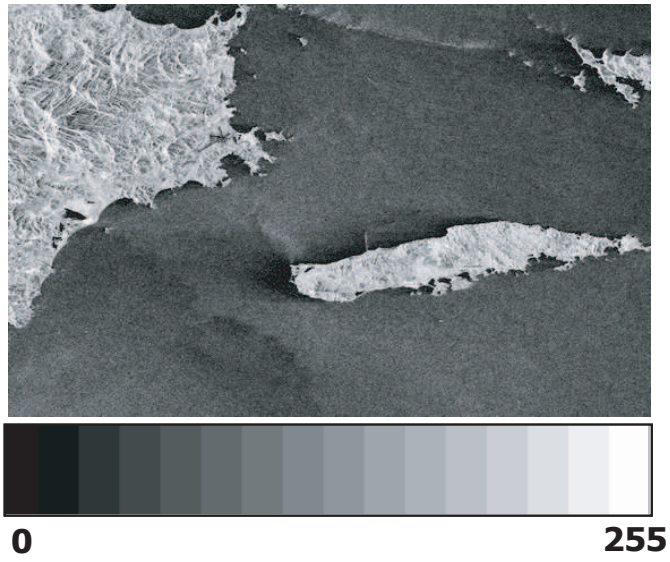

Figure 3.2: Grayscale Image

value. Only for display the gray values must be converted again to unsigned values by the inverse point operation":

$$q = (q' + 128)\, mod 256; -128 \leq q' \leq 128$$

3.3 Classification of Operators

This section presents a list of the implemented operators in JCID (Java Computational Image Developer). The classification was done according to the functionality of the operators.

3.3.1 Point Operators

These operators are used to modify the gray values at specific pixels. They can be applied to correct the image illumination and contrast enhancement.

- Absolute Value.
- Clamp.

- Color Convert.

3.3.2 Arithmetic Operators

This section includes the basic operations that could be done over images. It includes the operations of sum, substraction, multiplication, and division.

3.3.3 Spatial Operators

The function of these operators is to modify the position of the pixels.

- Crop. The operator cuts a portion of the image given the width and height of the section to be cropped.
- Resizing. The operator transforms the image by adding or substracting pixels.
- Flipping. The operator makes a mirror of the image in vertical or horizontal direction.

3.3.4 Convolution Operators

These operators are based on the convolution operation of image processing.

- Sharpening.
- Blurring
- Embossing
- Edge Detection

3.3.5 Filtering Operations

These operators are based also on the convolution operation and are mostly used for image enhancement, highlighting or hiding features of the image.

- Low-pass
- High-pass
- Laplacian
- Gaussian
- 2-D Fourier Transform

3.3.6 Complex Operators

These operators have been implemented especially for those images whose pixels are complex numbers.

- Cyclic Convolution
- Cyclic Correlation
- Conjugate
- Phase
- Hadamard Product
- Shifting

Chapter 4

CONNECTING MATLAB WITH JAVA

"***MATLAB***® is a high-level language and interactive environment that enables to perform computationally intensive tasks" [41]. It is very used into the scientific world because it provides a set of specialized functions that facilitates the programming work. Specialized functions includes, among others, signal and image processing, communications, control design, financial modeling and analysis, and computational biology.

The Image Processing Toolbox of ***MATLAB***® offers the following set of functions:

- Reading and Writing Image Data
- Display of Images
- Spatial Transformations: Interpolation, Resizing, Rotating, Cropping
- Image Registration
- Linear Filtering: supports linear convolution and linear correlation
- Filter Design
- Transforms: Fourier Transform, Discrete Cosine Transform

- Morphological Operations: Dilation, Erosion, Lookup Table Operations

- Analyzing and Enhancing of Images: Histograms, Edges Detection, Intensity Adjustment

In this project was considered the capability to use some of these functions into the proposed CIP system, in order to make reuse of code, and avoid the time consuming of programming. Kennedy et. al. [42] analyzed the issue of "compiling applications written in scripting languages (like "***MATLAB***®) into highly optimized machine code (C,C++,Java), to obtain a huge productivity advantage".

Following sections in this chapter shows three different approaches to connect ***MATLAB***® with Java.

4.1 Connection with JLab

JLab, is a Java library that allows calling ***MATLAB***® from Java code. This library was developed by Dr. Iain E. Toft [43]. JLab provides a set of methods to communicate the Java Virtual Machine (JVM) of the Java application with the JVM of the ***MATLAB***® Engine. Once is open the connection, an instance of the ***MATLAB***® Engine is created. That instance could receive ***MATLAB***® commands through the methods of the JLab library. When not any more commands has to entered, it must be closed the connection. Figure 4.1 illustrates the interaction between the two Java Virtual Machines. Figure 4.2 shows how is coded that interaction.

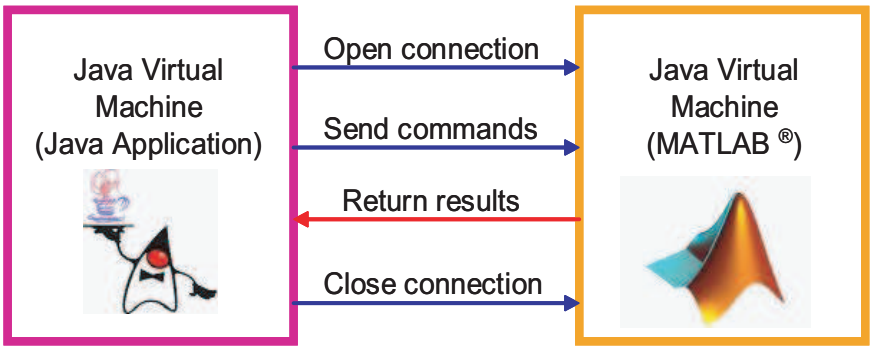

Figure 4.1: Interaction of the JVM between Java and MATLAB using JLab

4.2 Connection with Sockets

"A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program."[44]

4.3 Connection with JavaBuilder

According to the approach presented by [42], *MathWorks* released in its last version of ***MATLAB***® (R2006b), a new toolbox called **javabuilder**, which attempts to encapsulate ***MATLAB***® files into Java classes that can be then invoked by a Java application. Figure 4.4 shows an screenshot of the environment of this new tool.

With this new mechanism, it is very attractive to scientists develop its own algorithms in ***MATLAB***® without worry about develop it with Java. Java Builder provides all the capabilities necessary to translate the array structures and data types of ***MATLAB***® to Java. User only have to develop the Java application that calls the Java classes generated by Java Builder.

```
MatlabEngine engine = MatlabEngine.getEngineInstance();
        try {
                engine.openEngine();
                engine.putDoubleMatrix("doubmat", preal);
                engine.evalString("imshow(doubmat);");

        engine.evalString("imwrite(doubmat,'C:/Hadamard.jpg');");
                try {
                        Thread.sleep(5000);
                }
                catch(InterruptedException e) {
                        e.printStackTrace();
                }
        }
        catch(MatlabEngineException mle)
        {
                mle.printStackTrace();
                System.out.println(mle);
        }
        finally
        {
                try {
                        engine.closeEngine();
                }
                catch(MatlabEngineException mle)
                {
                        mle.printStackTrace();
                        System.out.println(mle);
                        System.exit(-1);
                }
        }
```

Figure 4.2: Code of the interaction between Java and MATLAB

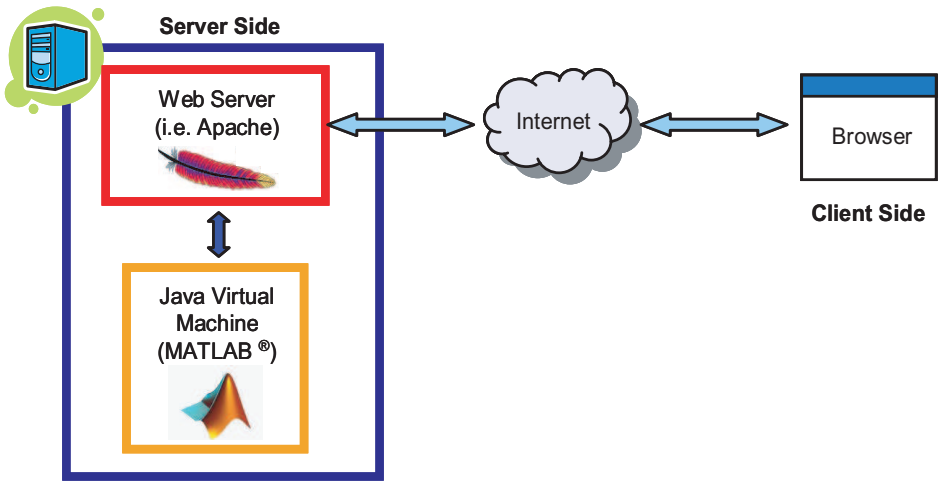

Figure 4.3: Communication with Sockets

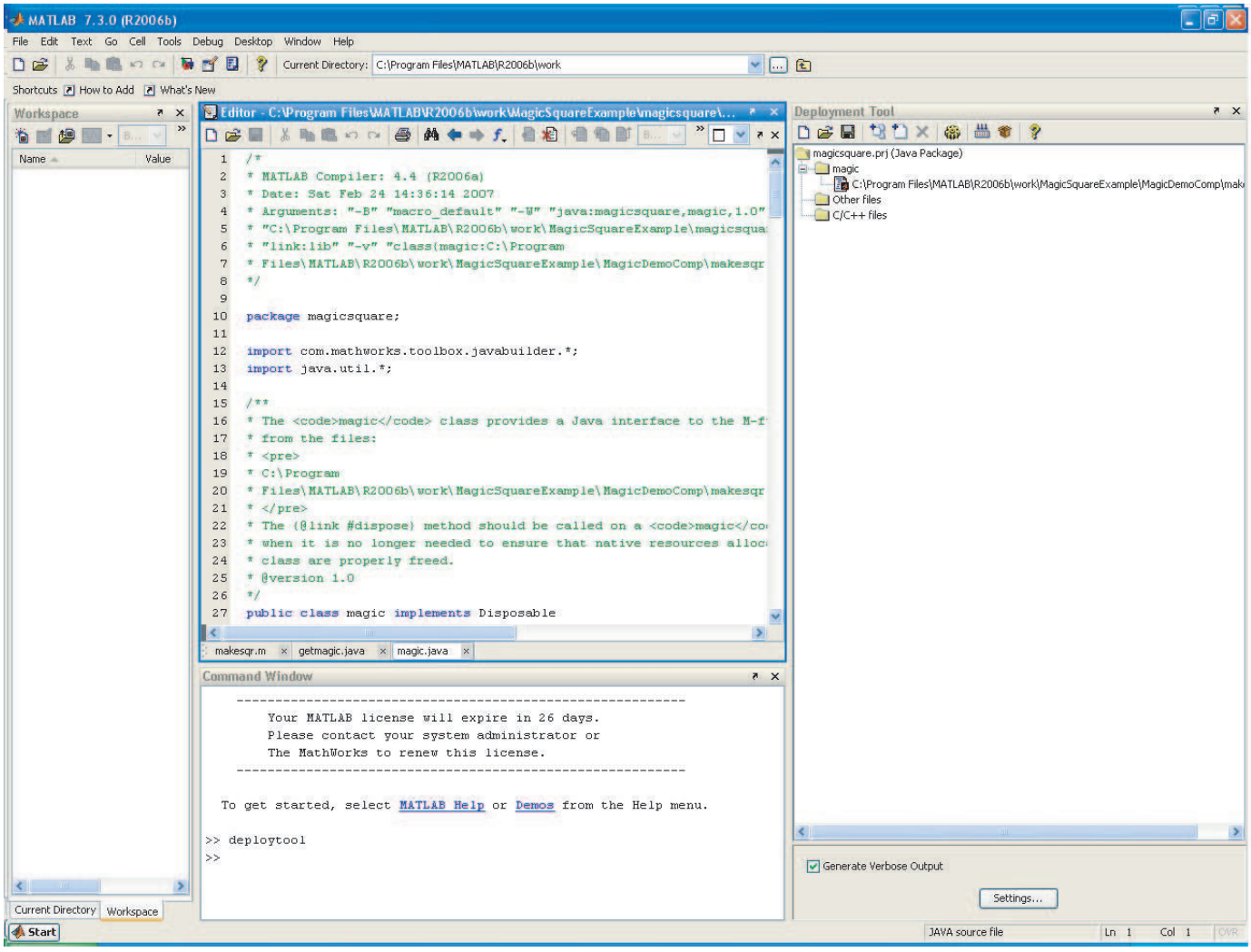

Figure 4.4: Screenshot of the Javabuilder Deploy Tool

Chapter 5

WEB-BASED COMPUTATIONAL SIGNAL PROCESSING

In Chapter 2 the mathematical fundamentals were presented from where the computational signal processing system had been built. This chapter describes how the mathematical structures were implemented over computational structures, using the programming language Java as tool for the coding of the algorithms. Also a theoretical background about web-based applications is presented.

5.1 Software Tools used for Implementation

The implementation of all the components of the system was done using the programming language Java. It was chosen because the following features [45]:

- It is object-oriented, which means that the programming of the state and behavior of a system is done by programming the state and behavior of the objects that compounds the system.
- It is platform-independent, or capable to run on different platforms such as Windows,

Linux, Macintosh.

- It is the most common programming language for web applications, because it is the foundation of many developing frameworks.

Between the several development projects of the Java community, there are two important APIs specialized for treatment of images. The first is the **Java Advanced Imaging API**. This is a set of interfaces to manipulate images. The second is **Java Image I/O API**, which provides management of image files stored in a local file system or distributed across the network [45].

Considering that was assumed that the input data are complex signals, it means, each sample is a complex number (formed by a real number and an imaginary number), it was acquired an open source library of operations that supports complex numbers, called *Flanagan's Java Library* (flanagan.jar), created by Dr. Michael Thomas Flanagan from the University College London. Operations includes complex arithmetic: addition, subtraction, multiplication, and division. Also includes trigonometric operations and special mathematical functions.

For the implementation of the web environment was chosen **Java Server Faces**, most known as *JSF*. JSF is a project from the Java Community Process [46]. JSF applications are standard Java Web Applications, for which JSF defines three layers: a component architecture which defines a common way to build user interface (UI) widgets, a standard set of UI widgets (such as text boxes, list boxes, tabbed panes, and data grids), and an application infrastructure which uses HTTP protocol for communication via the Servlet API, uses JavaServer Pages (JSP) as display technology, and uses JavaBeans for exposing properties and event handling [47].

There exists many distributions of JSF from different companies such as Oracle, IBM and Apache Project. In this work the open source implementation of JSF developed by Apache Project, called **MyFaces** was used.

Apache Tomcat was used as servlet container, that is used in the "official Reference Implementation for the Java Servlet and JavaServer Pages technologies, providing an environment for Java code to run in cooperation with a web server". Tomcat is cross-platform, running on any operating system that has a Java Runtime Environment.

JFree Chart is a free Java chart library that makes it easy for developers to display professional quality charts in their applications. It consists of an API that supports a wide range of chart types, a flexible design that is easy to extend, and targets both server-side and client-side applications. It is distributed under the terms of the GNU Lesser General Public License (LGPL), which permits use in proprietary applications.

5.2 Object-Oriented-Based Signal Operator Approach

Object-Oriented programming (OOP) is a programming paradigm that uses "objects" to design applications. "An object is a kind of self-sufficient entity that has an internal *state* (the data it contains) and that can respond to *messages* (calls to its subroutines). The OOP approach to software engineering is to start by identifying the objects involved in a problem and the messages that those objects should respond to. The program that results is a collection of objects, each with its own data and its own set of responsibilities. The objects interact by sending messages to each other" [48].

Object-Oriented-Based Signal Operator Approach (OOSO) is a methodology to model

aspects of the Computational Information Processing (CIP) Framework, which comprises the set of inputs, the set of outputs, and the rules of composition.

5.2.1 Data Input Structures

The CSP system has two basic structures of data input, they are: one-dimensional (1-D) signals of dimension $\boldsymbol{N}$ and two-dimensional (2-D) signals of dimension $\boldsymbol{M} \times \boldsymbol{N}$. A 1-D signal is represented mathematically as a sequence of samples like this:

$$\boldsymbol{x}\left[\boldsymbol{n}\right] = \left\{\boldsymbol{x}\left[\mathbf{0}\right], \boldsymbol{x}\left[\mathbf{1}\right], \boldsymbol{x}\left[\mathbf{2}\right], \ldots, \boldsymbol{x}\left[\boldsymbol{N}-\mathbf{1}\right]\right\}$$

Graphically, this sequence can be seen as 1-D array, or as a *vector* data structure:

Figure 5.1: 1-D signal represented as vector

A 2-D signal is represented mathematically as a matrix of samples like this:

$$\boldsymbol{x}\left[\boldsymbol{n_x}, \boldsymbol{n_y}\right] = \left\{\begin{array}{ccccc} \boldsymbol{x}\left[\mathbf{0},\mathbf{0}\right] & \boldsymbol{x}\left[\mathbf{0},\mathbf{1}\right] & \boldsymbol{x}\left[\mathbf{0},\mathbf{2}\right] & \ldots & \boldsymbol{x}\left[\mathbf{0},\boldsymbol{M}-\mathbf{1}\right] \\ \boldsymbol{x}\left[\mathbf{1},\mathbf{0}\right] & \boldsymbol{x}\left[\mathbf{1},\mathbf{1}\right] & \boldsymbol{x}\left[\mathbf{1},\mathbf{2}\right] & \ldots & \boldsymbol{x}\left[\mathbf{1},\boldsymbol{M}-\mathbf{1}\right] \\ \boldsymbol{x}\left[\mathbf{2},\mathbf{0}\right] & \boldsymbol{x}\left[\mathbf{2},\mathbf{1}\right] & \boldsymbol{x}\left[\mathbf{2},\mathbf{2}\right] & \ldots & \boldsymbol{x}\left[\mathbf{2},\boldsymbol{M}-\mathbf{1}\right] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}\left[\boldsymbol{N}-\mathbf{1},\mathbf{0}\right] & \boldsymbol{x}\left[\boldsymbol{N}-\mathbf{1},\mathbf{1}\right] & \boldsymbol{x}\left[\boldsymbol{N}-\mathbf{1},\mathbf{2}\right] & \ldots & \boldsymbol{x}\left[\boldsymbol{N}-\mathbf{1},\boldsymbol{M}-\mathbf{1}\right] \end{array}\right\}$$

Graphically, this can be seen as a 2-D array data structure as is presented in Figure 5.2.

Making an extension to OOP, both the 1-D and the 2-D signals represent each one a

$x[n_x, n_y] =$

x [0,0]	x [0,1]	x [0,2]	...	x [0,M-1]
x [1,0]	x [1,1]	x [1,2]	...	x [1,M-1]
x [2,0]	x [2,1]	x [2,2]	...	x [2,M-1]
⋮	⋮	⋮	⋱	⋮
x [N-1,0]	x [N-1,1]	x [N-1,2]	...	x [N-1,M-1]

$n_x = \{0,1,2,\ldots N-1\},\ n_x \in \mathbb{Z}_N$

$n_y = \{0,1,2,\ldots M-1\},\ n_y \in \mathbb{Z}_M$

Figure 5.2: 2-D signal represented as matrix

class with its own attributes and methods. Figure 5.3 illustrates the attributes that identify each class.

For 1-D signals the class has been called `Signal`. The attribute `complexSignal` of type `Complex[]` is a vector that contains the complex samples read from the data file. If the data are just real numbers ($\mathbb{R}$), the imaginary part of each sample of the complex array is set to zero. Each element of the array is an object of the class `Complex` from the Flanagan's jar library. Figure 5.4 shows the class diagram of the `Complex` class. For each loaded signal there is a `List` of all the transformations resulting from the action of the operators over the signal. This list is represented by the attribute `transformedList` of the `Signal` class.

The class of 2-D signals has been called `ImageJCID`. The attribute of type `RenderedImage` represent the image as a grid of pixels. Images in format .jpg, .gif, .bmp, and .tiff are easily loaded in this type of data. With this representation is possible to get the values of the pixels as 2-D arrays, with `int` data type or as a matrix of complex numbers of type `Complex[][]`. As in the 1-D case, each image has its corresponding `List` of images that result of the action

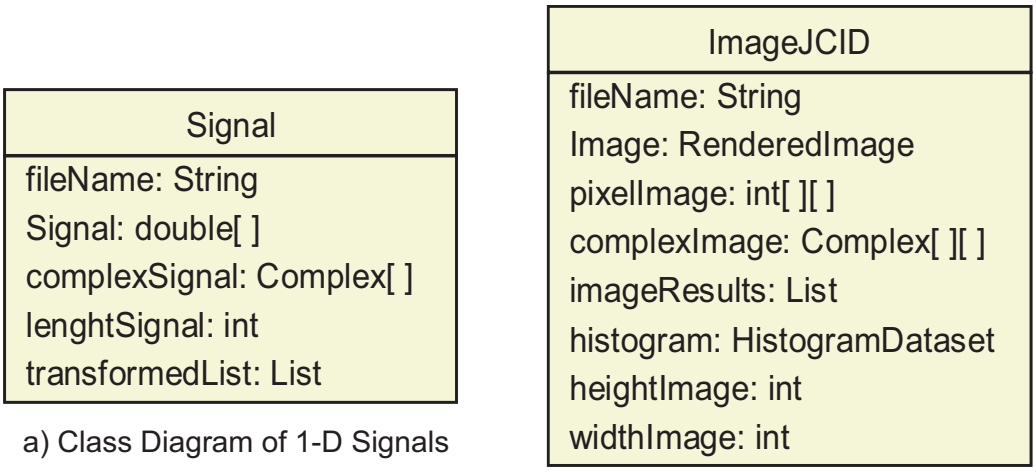

Figure 5.3: Class Diagrams of 1-D and 2-D signals

of the operators.

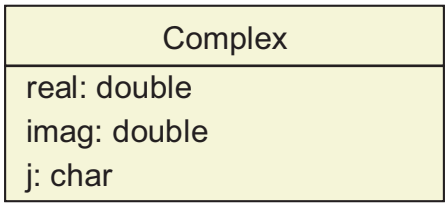

Figure 5.4: Class Diagrams of Complex class

5.2.2 Operator Structures

Operators are systems able to transform an input signal to produce an output signal. For the CIP, they were classified in unary operators and binary operators. Unary operators take only one signal on the input and produce one signal on the output, while binary operators take two signals on the input and produce one signal on the output. Figure 5.5 shows an example of an unary operator like the Discrete Fourier Transform, and an example of a binary operator like the addition.

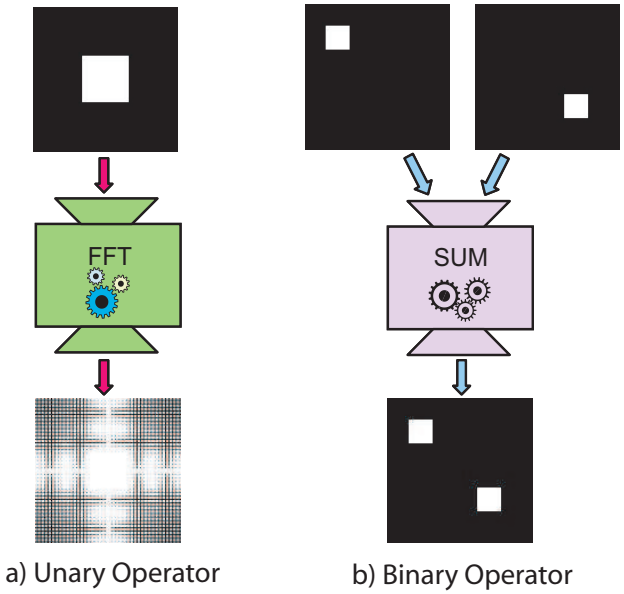

Figure 5.5: Example of Unary and Binary Operators

Such distinction of unary and binary operators apply to both 1-D signals and 2-D signals. Figure 5.6 is the class diagram for both kind of operators. The set of implemented operators is classified in operators to transform 1-D signals, called `OneDimOperators`, and operators to transform 2-D signals called `TwoDimOperators`. Each division is classified in `UnaryOperator` and `BinaryOperator`, which in turn, are composed by all the different operators that fit to each category.

Now, in Figure 5.7 it can be seen the relationship of transformation of the Operators classes over the Signal classes.

5.3 Web Application Architecture

A web application could be defined as a web system where user input (navigation and data input) affects the state of the logic of the system. The basic architecture of a web application includes browsers, a network, and a web server. In reference [49], Jim Conallen states also that "a web application uses a web site as the front end to a more typical application".

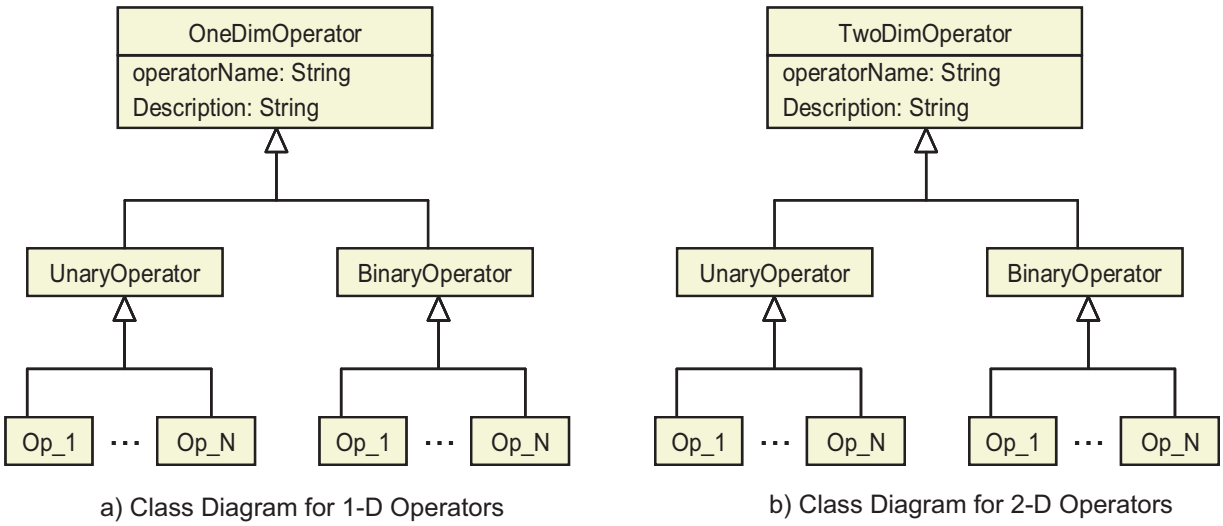

Figure 5.6: Class Diagrams for Operators

Figure 5.8 shows the relationship between the main components of a web application.

The connection between the client and server only exists during a page request. Once the request is complete, the connection is broken. All the activity on the server occurs during the page request [49].

The client/server architecture has been one of the most used network architectures to build web applications. In this architecture, the client is the requester of services and the server is the provider of such services. The client could be interpreted as a desktop application installed on a several number of workstations, which make requests to a single application installed on the server. This kind of interaction is known as *two-tiered architecture*.

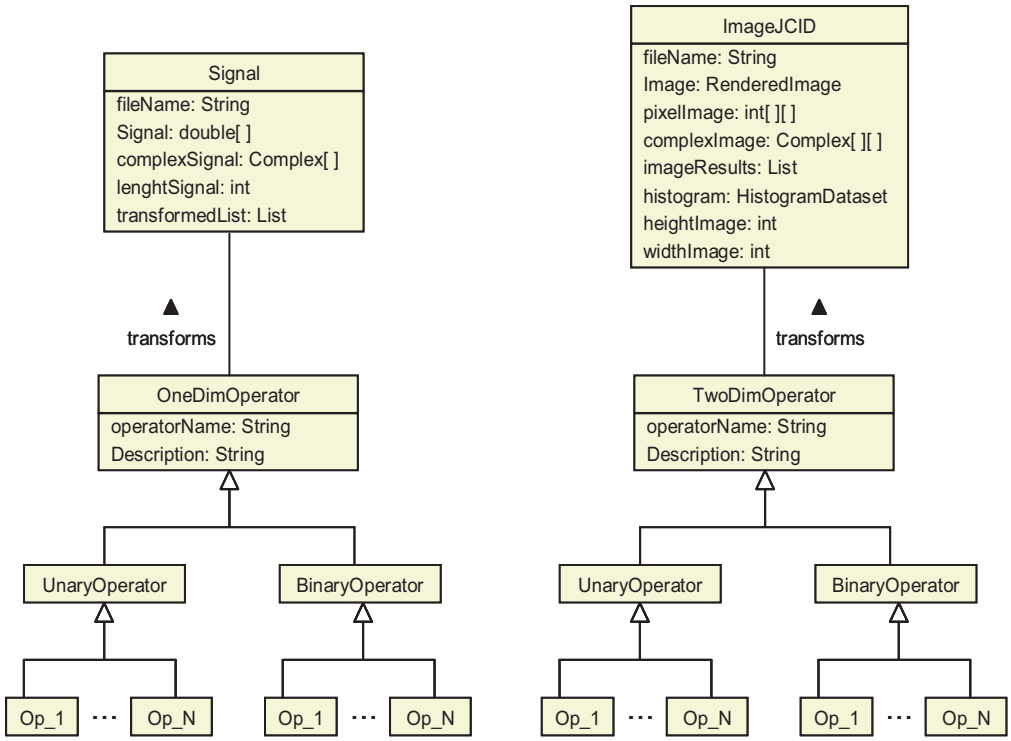

Figure 5.7: Class Diagram of the Action of Operators over Signals

5.4 Multitiered Applications

Though many variations are possible, a web application is commonly structured as a three-tiered application, which is the most common example of multi-tier architecture. In its most common form, a web browser is the first tier, an engine created using some dynamic web content technology (e.g., CGI, PHP, Java Servlets or Active Server Pages) is the middle tier, and a database is the third tier. The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generating a user interface. Figure 5.9 shows the interaction between the three elements of this architecture.

In three-tier applications the Presentation tier never communicates directly with the Data tier. Additional to this configuration, another component has been added by the **Model-View-Controller (MVC)** design pattern. This pattern insert an intermediate

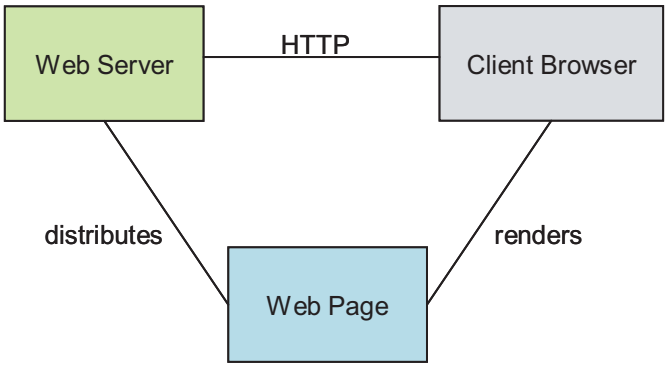

Figure 5.8: Components of a Web Application

component called the **controller**, which is between the Presentation tier and the Logic tier. The function of the controller is to process and respond to events, typically user actions, and invoke changes on the model [50]. Figure 5.10 illustrates the standard Model-View-Controller Architecture.

In section 5.1 was explained that JavaServer Faces was chosen to implement this architecture for the web-based computational signal processing. This development framework uses also the **J2EE (Java Platform, Enterprise Edition)** framework of Java Sun Microsystems. "The J2EE platform offers a multitiered distributed application model, reusable components, a unified security model, flexible transaction control, and web services support through integrated data interchange on Extensible Markup Language (XML)-based open standards and protocols" [50].

In the context of the web-based computational signal processing system, the basic structures of the set of multidimensional signals and the set of implemented operators reside in the Model tier of the MVC model. They are invoked by the `FacesServeletContext` instance of the Controller tier. Next chapter explains in more detail how was developed each one of the modules that compose the environment.

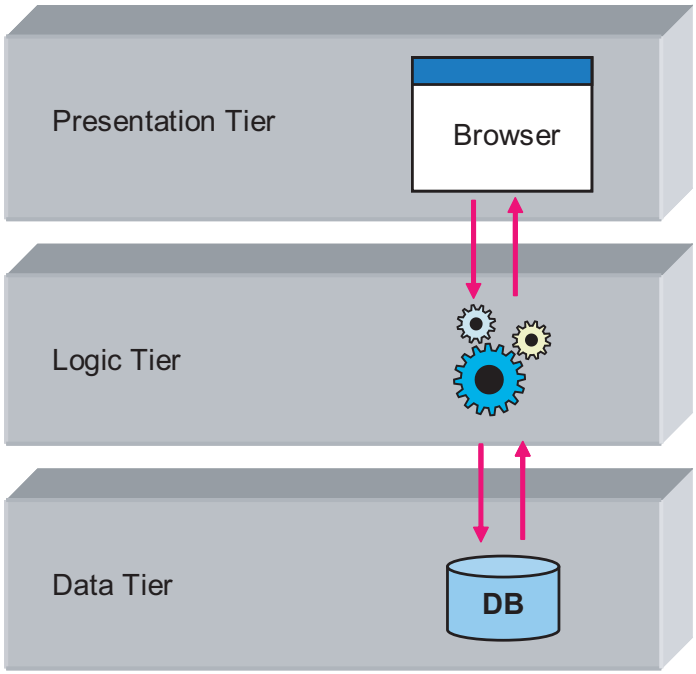

Figure 5.9: Three-Tier Architecture

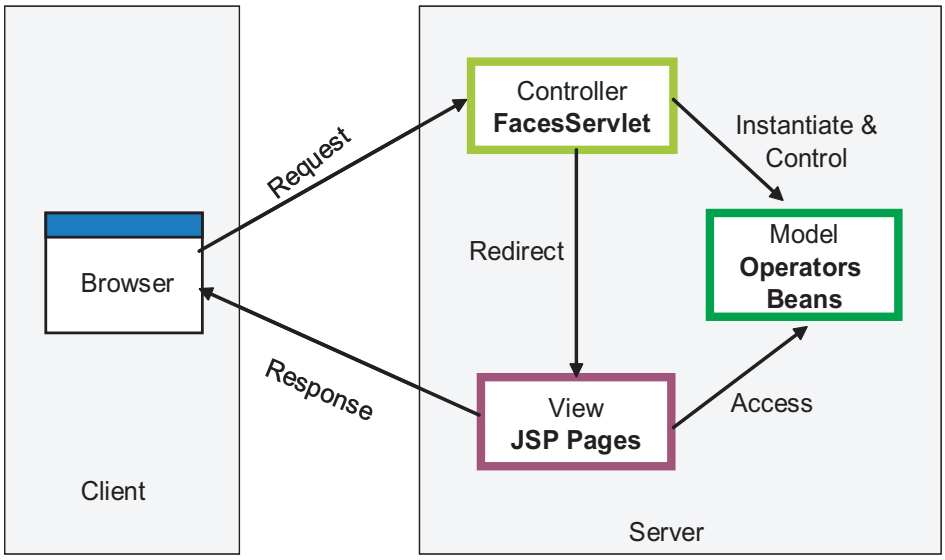

Figure 5.10: Model-View-Controller Architecture

Chapter 6

JCID: JAVA COMPUTATIONAL IMAGE DEVELOPER

In order to demonstrate the functionality of the theory presented in the two last chapters, it was necessary to implement an application that could help to scientists that work with multidimensional signals. This application was named **JCID: Java Computational Image Developer**. In first instance, it was thought that this tool would process only 2-D signals, specifically signals that could be represented as images, but in an advanced stage of the development of the project, it was considered the alternative to process also 1-D signals, since some raw data could be available to the research group.

This application is part of the general project "**WALSAIP: Wide Area Large Scale Automated Information Processing**" of the Institute for Computing and Informatics Studies of the Department of Electrical and Computer Engineering. In this project participate the following research groups:

- Advanced Data Management Group (ADMG)
- Automated Information Processing Group (AIPG)

- Human Computer Interfaces Group (HCIG)
- Hydro-Ecological Research Group (HERG)
- Network Communication Infrastructure Group (NCIG)
- Parallel and Distributed Computing Group (PDCG)

This project have the following technological infrastructure, composed by a set of 7 servers, each one with 28 disks of 300 Gigabytes. Each one of the research groups have one of the servers. The infrastructure belongs to the network core of INEL/ICOM department, which acts as an intermediate between the internal network and the exterior world (Internet and another dependencies of the UPRM). Figure 6.1 is an scheme of this infrastructure.

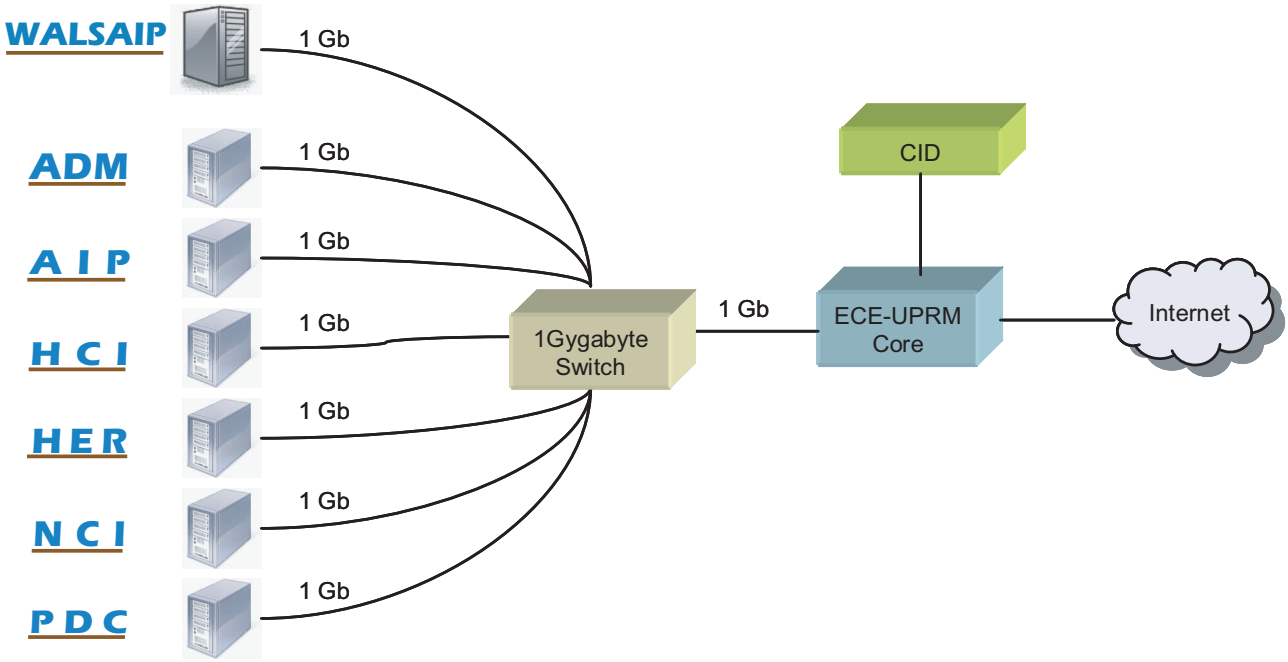

Figure 6.1: Infrastructure of WALSAIP Project

The WALSAIP project has an agreement with the Jobos Bay Reserve, a dependency of NOAA located in the municipality of Salinas in Puerto Rico, dedicated to the monitoring of the hydrological and ecological factors of the coastal zone. In that reserve had been set some environmental sensors that are collecting information which is being stored in a database. Figure 6.2 shows a photography of the area of the Jobos Bay.

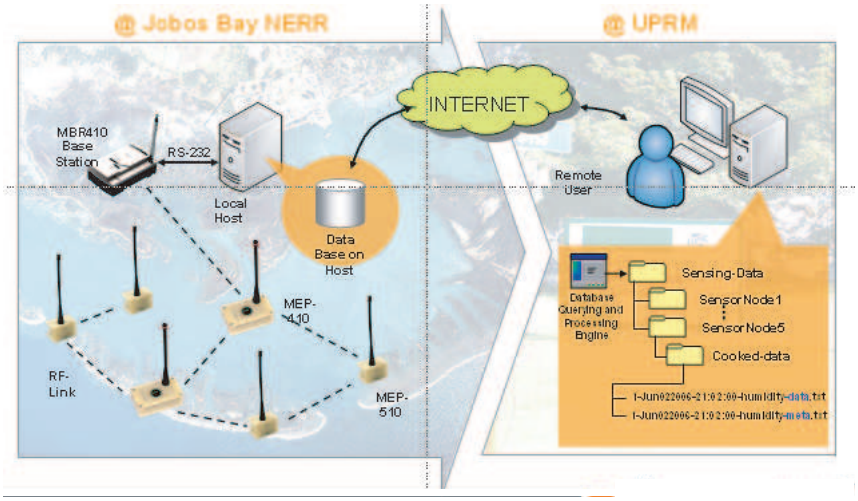

Figure 6.2: Jobos Bay Reserve

The development of the application was done in two phases: first was developed a stand-alone version in order to make a "prove of concept" of the implemented operators and the integration in a graphical user interface. The second phase was the implementation as web application making use of the network infrastructure described above.

In the design of the application was taking into account some rules of Usability Engineering and Human Computer Interaction in order to provide a user friendly interface that facilitates the work of the scientists. Figure 6.3 is an screenshot of the graphical user interface of JCID.

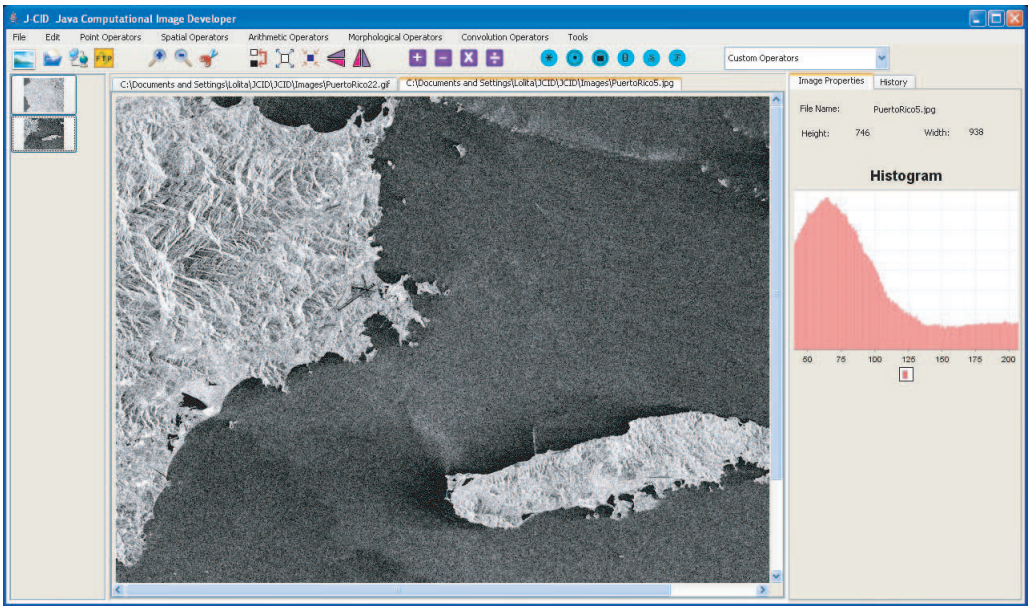

Figure 6.3: Graphical User Interface of JCID

In the menu bar of JCID appears the available operators to process the signals. This operators were classified by the type of functionality, for instance, to perform arithmetic operations over the signals, there is a menu of arithmetic operators, like addition, substraction, multiplication, division, exponential and logarithmic. If user is working with an image and wants to enhance that image, there is a menu of filtering operators, like edges detection, low pass filter, high pass filter, etc.

User can apply a sequence of operators over the signal, and after get an optimal result, that sequence can be encapsulated in a new operator, in order to avoid the repetition of steps in another session. This helps to user make a repository of his/her own operators. Figure 6.4 illustrates how is presented the sequence of actions to the user.

In the right panel of the figure appears the sequence of operators represented as thumbnails of the obtained results. At the end of the sequence there is a command button to execute the encapsulation of the sequence. In Figure 6.5 can be seen the window dialogs

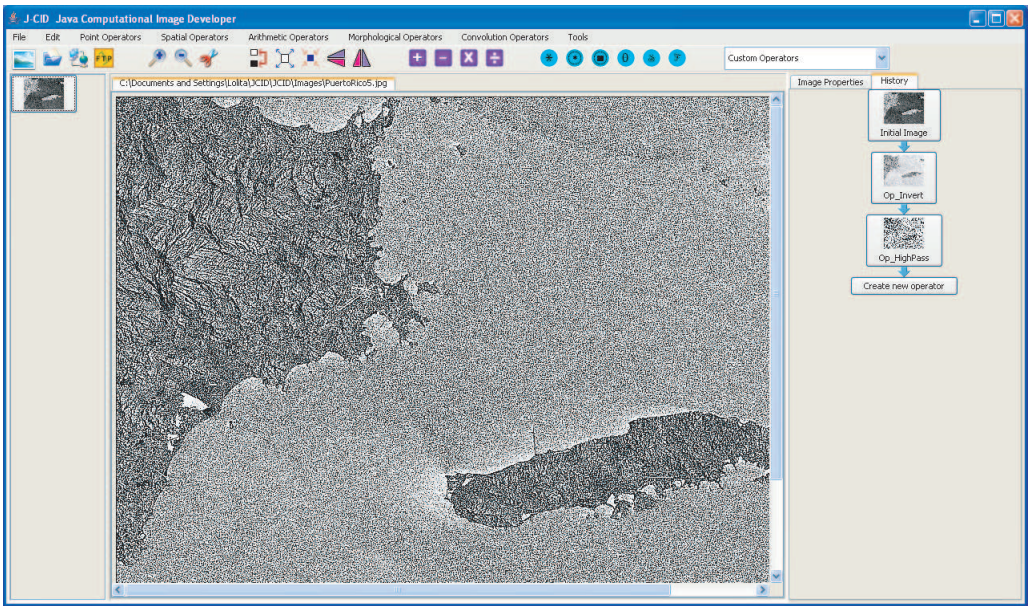

Figure 6.4: Encapsulation of Operators

where appears the list of operators that user wants to encapsulate, and a small editor if the user wants to add more functionalities to the new operator.

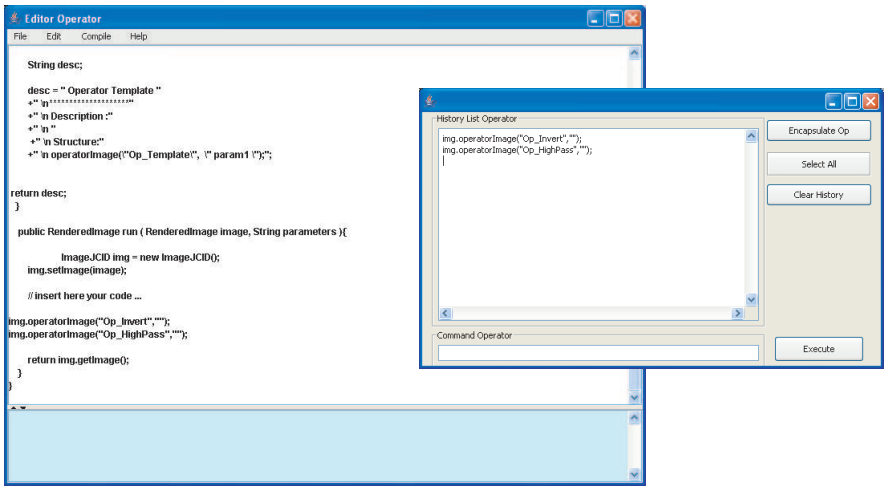

Figure 6.5: Edition and Compilation of new Operators

The operators of Dilation and Erosion of images that appears in the Morphological Operators menu were implemented using the connection between ***MATLAB***® and Java using

the JLab library, which was described in section 3.2. The operator of the Short Time Fourier Transform (STFT) was also deployed with the Javabuilder toolbox. This was made by using a trial version of the toolbox. When the license of the toolbox could be acquired, it could be implemented more operators for future versions of the application.

Between another functionalities of JCID, there is an interface to connect to the database that collects the data from the sensors. This interface helps to make queries according to the time interval the user wants to analyze. Results of the queries are stored in files of data and metadata. Data files contain the measures of the sensors, and metadata files contain the information related to the sensor, its location, and the settings for the sampling of the data. Figure 6.6 shows an sreenshot of the interface.

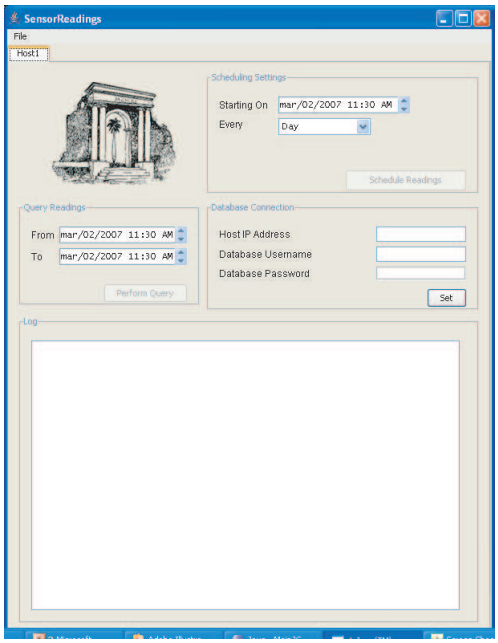

Figure 6.6: Connection with the sensors Database

Since was considered that user could have data files in another servers o workstations in the network, into JCID was integrated an open source application that allows to open an

FTP connection. This application is called **J-FTP** developed by *JMethods Inc.* Figure 6.7 is an screenshot of J-FTP.

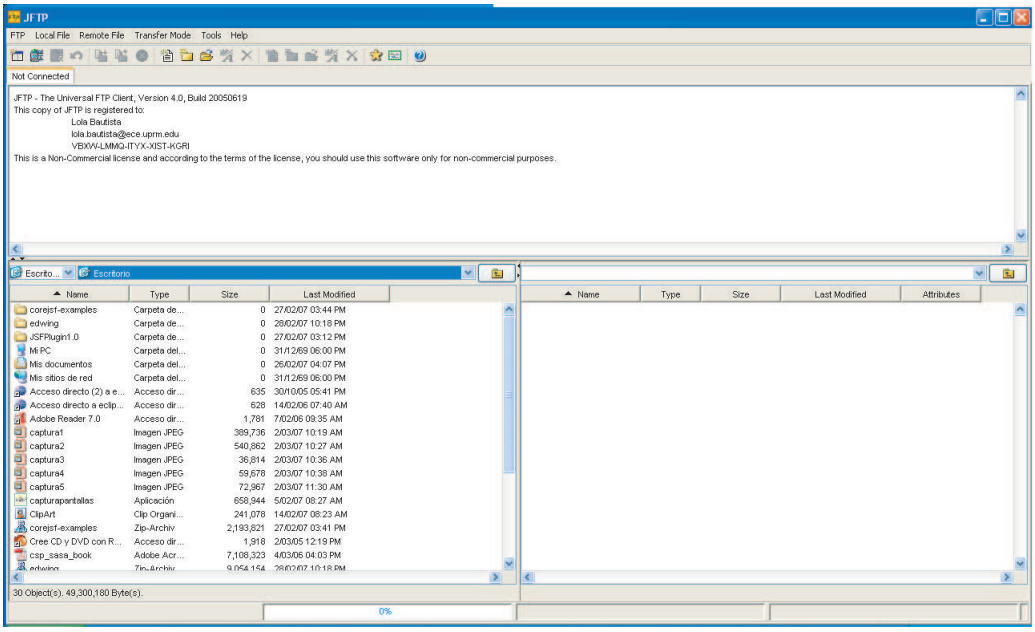

Figure 6.7: FTP connection using J-FTP

The web version of JCID has been deployed in the WALSAIP server of the project. It contains the same functionalities that were developed in the stand-alone version. In order to conserve the same graphical configuration of the stand-alone, the user interface of the web version was coded with JavaServer Faces using the tag libraries released in the MyFaces distribuition of Apache Project. Figure 6.8 is a display of the web version.

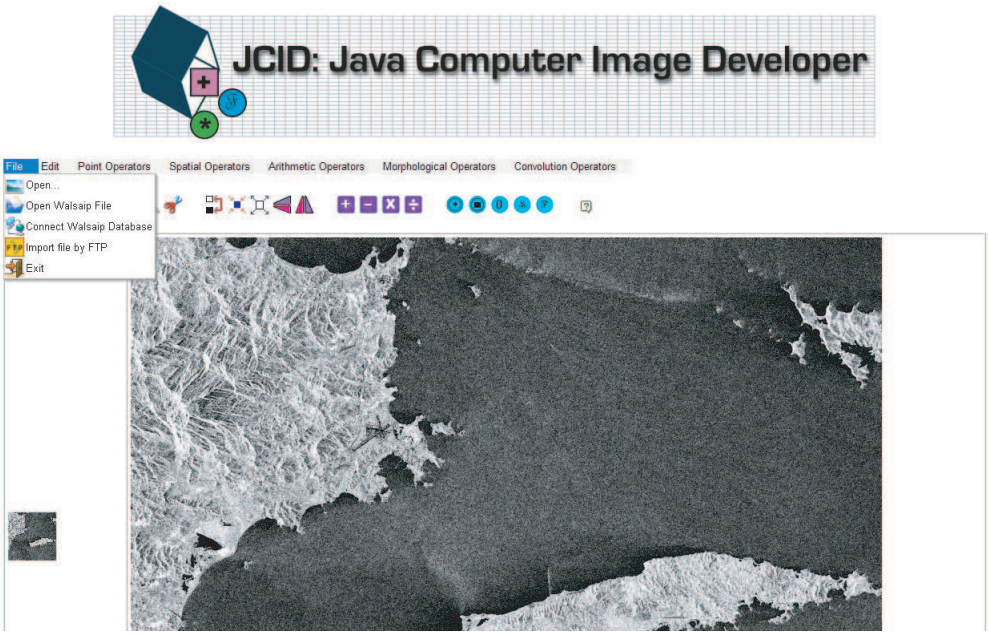

Figure 6.8: Graphical User Interface of Web-JCID

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The Computational Signal Processing Environment has been designed to be a useful tool for geologists, hydrologists, and scientists in related areas. It provides a friendly access to basic and advance operators, with encapsulation capabilities.

One of the most important features of the environment is the portability offered by Java implementation, that also facilitates the availability to deploy the source code for potential users and developers.

Integration of Java with scripting languages like ***MATLAB***® enhance the time consuming in coding and provides a better tool for scientists who does not have experience in programming languages.

7.2 Future Work

In order to contribute to create a community for web-based signal processing, it is taking into account the possibility to distribute the application to different nodes that belongs to the project **PlanetLab**, which is a consortium of academic, industrial, and government institutions to share resources to process large amounts of data [51].

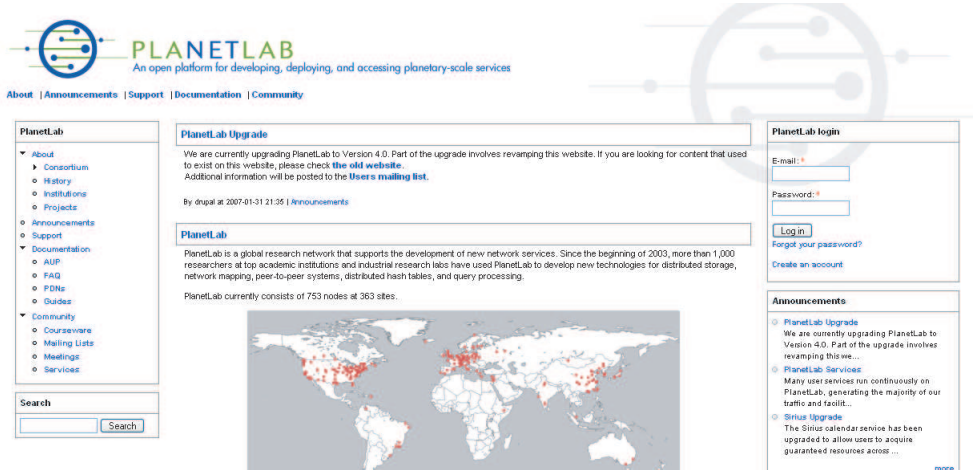

Figure 7.1: PlanetLab Home Page

GENI is another project where could be demonstrated the utility of JCID. GENI is a project of the National Science Foundation whose goal is obtain a global communications network [52].

Other enhancement that could be done on JCID is to add capabilities to integrate with GIS tools, in order to process georeferenced data that is widely used by hydrologists.

Figure 7.2: GENI Project Home Page

Bibliography

[1] NOAA's Office of Satellite Operations. http://www.oso.noaa.gov/history.

[2] NASA's Destination Earth Program. http://www.earth.nasa.gov/history/index10.html.

[3] Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Earth _ science.

[4] Warren Viessman Jr.; Gary L. Lewis; John W. Knapp. *Introduction to Hydrology*, page Chapter 1. Addison-Wesley, 1989.

[5] Open Geospatial Consortium Inc. *Sensor Model Language (SensorML) for In-situ and Remote Sensors*, 2004.

[6] Robert H. Shumway; David S. Stoffer. *Time Series Analysis and Its Applications with R Examples*, pages 4–11. Springer, 2000.

[7] Domingo Rodriguez. Technical report nsf. Technical report, UPRM, 2006.

[8] Corina Alecu; G. Stancalie. Use of geographic information and remotely sensed data in the decision-making support system for flood management in romania. *Proceedings of the 2nd International Conference on Recent Advances in Space Technologies, RAST 2005*, pages 583–587, 2005.

[9] R.G. Bryant; M.P. Rainey. Investigation of flood inundation on playas within the zone of chotts, using a time-series of avhrr. *Elsevier, Remote Sensing of Environment*, 82:360–375, 2002.

[10] Frédéric Frappart; Frédérique Seyler; Jean-Michel Martinez; Juan G. León; Anny Cazenave. Floodplain water storage in the negro river basin estimated from microwave remote sensing of inundation area and water levels. *Elsevier, Remote Sensing of Environment*, 99:387–399, 2005.

[11] Seng Chuan Tay; Kim Hwa Lim. Web-based processing for remotely sensed data. *IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2002*, 4:2531–2533, 2002.

[12] Karl Crowley; June Frisby; S éamus Murphy; Mark Roantree; Dermot Diamond. Web-based real time temperature monitoring of shellfish catches using a wireless sensor network. *Elsevier, Sensors and Actuators*, 122:222–230, 2005.

[13] G.X. Ritter. Image algebra. Technical report, Department of Computer and Information Science and Engineering University of Florida, 1993, 1999.

[14] I.G. Angus. Image algebra: An object oriented approach to transparently concurrent image processing. *Proceedings on Scalable High Performance Computing Conference, SHPCC 1992*, pages 9–13, 1992.

[15] Reiner Lenz; Thanh Hai Bui; Koichi Takase. A group theoretical toolbox for color image operators. *IEEE International Conference on Image Processing, ICIP 2005*, 3:557–560, 2005.

[16] Christian Kiehle. Business logic for geoprocessing of distributed geodata. *Elsevier, Computer and Geosciences*, 32:1746–1757, 2006.

[17] C. Hill; C. DeLuca; Balaji; M. Suarez; A. DaSilva. The architecture of the earth system modeling framework. *IEEE Computational Science and Engineering*, 6:18–28, 2004.

[18] R.L. King; R.J. Birk. Developing earth system science knowledge to manage earth's natural resources. *IEEE Computational Science and Engineering*, 6:45–51, 2004.

[19] Paolo Diviacco. An open source, web based, simple solution for seismic data dissemination and collaborative research. *Elsevier, Computers and Geosciences*, 2005.

[20] Plale B.; Gannon D.; Alameda J.; Wilhelmson B.; Hampton S.; Rossi A.; Droegmeier K. Active management of scientific data. *IEEE Internet Computing*, 9:27–34, 2005.

[21] J. Nogueras-Iso; F. J. Zaragoza-Soria; R. Bejar; P. J. Alvarez; P. R. Muro-Medrano. Ogc catalog services; a key element for the developmet of spatial data infrastructures. *Elsevier - Computer and Geosciences*, 31:199–209, 2005.

[22] Center for Advanced Computing Research of the California Institute of Technology. Sar atlas project. http://www.cacr.caltech.edu/Publications/annreps/annrep95/compsci1.html.

[23] Geosciences network (geon). http://www.geongrid.org/about.html.

[24] Infovis cyberinfrastructure, a data-code-compute for research and education in information visualization. http://iv.slis.indiana.edu/sw.

[25] R. de Beer; C. Couturier; D. Graveron-Demilly; S. Nastase; D. van Ormondt; E.D. Riedijk. Java-based visual information system for image reconstruction algorithms. *Proceedings of the 2nd STW Workshop on Semiconductor Advances for Future Electronics, SAFE 99*, pages 27–32, 1999.

[26] R. de Beer; D. Graveron-Demilly; S. Nastase; D. van Ormondt. A distributed computing system for magnetic resonance imaging: Java-based processing and binding of xml. *Elsevier, Computer Methods and Programs in Biomedicine*, 73:221–231, 2004.

[27] George K. Matsopoulos; Vassilis Kouloulias; Pantelis Asvestas; Nikolaos Mouravliansky; Konstantinos Delibasis. Mitis: a www-based medical system for managing and processing gynecological-obstetrical-radiological data. *Elsevier, Computer Methods and Programs in Biomedicine*, 76:53–71, 2004.

[28] A. J. Page; T. M. Keane; and T. J. Naughton. Bioinformatics on a Heterogenous Java Distributed System. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposyum*, 2005.

[29] E. S. Manolakos; D. G. Galatopoullos; and A. P. Funk. Distributed matlab based signal and image processing using javaports. *ICASSP IEEE*, 5:17–21, 2004.

[30] C. Hualparimachi; D. Rodriguez. Java-based tool for synthetic aperture radar image analysis. *Proceedings of the 3rd International Symposium on Image and Signal Processing Analysis*, 2003.

[31] Jonathan Campbell; Fionn Murtagh; M ünevver K ök üer. Datalab-j: A signal and image processing laboratory for teaching and research. *IEEE Transactions on Education*, 44:329–335, 2001.

[32] Andreas Spanias; Fikre Bizuneh. Development of new functions and scripting capabilities in java-dsp for easy creation and seamless integration of animated dsp simulations in web courses. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001*, 5:2717–2720, 2001.

[33] D. Rodriguez. Computational signal processing and sensor array signal algebra, 2002.

[34] Dan E. Dudgeon; Russell M. Mersereau. *Multidimensional Signal Processing*, page 12. Prentice-Hall Inc., 1984.

[35] Sanjit K. Mitra. *Digital Signal Processing: A Computer-Based Approach*, pages 71–76. McGraw-Hill, 2001.

[36] A.; Sell G. Naylor. *Linear Operator Theory in Engineering and Science*, pages 17–21. Holt, Rinehart and Winston, Inc., 1971.

[37] Burrus C.S. *Algebraic Methods for Signal Processing and Communications Coding*, pages 7–15. Springer-Verlag, 1992.

[38] W. Greub. *Linear Algebra*, pages 5–37. Springer-Verlag, 1981.

[39] Jähne B. *Digital Image Processing*. Springer, 2005.

[40] Gonzalez R.; Woods R. *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[41] Mathworks. Matlab, the language of technical computing. http://www.mathworks.com/products/matlab/.

[42] Kennedy K.; Broom B.; Chauhan A.; Fowler R.;Garvin J.;Koelbel C.;McChosh C.;Mellor-Crummey J. Telescoping languages: A system for automatic generation of domain languages. *Invited Paper, Proceedings of IEEE*, 93:387 – 408, 2005.

[43] I. E. Toft. JLab - Connecting Java to MATLAB http://www.cmp.uea.ac.uk/ it/jlab/index.html. University of East Anglia School of Computing Sciences, Norwich, Norfolk, U.K. NR4 7TJ, 2005.

[44] All about Sockets. http://java.sun.com/docs/books/tutorial/networking/sockets/index.html.

[45] Java Technology. http://www.sun.com.

[46] Java community process. http://jcp.org/en/home/index.html.

[47] K. Mann. *JavaServer Faces in Action*, pages 16–37. Manning Publications Co., 2005.

[48] David J. Eck. Introduction to programming using java. Hobart and William Smith Colleges, http://math.hws.edu/javanotes, 2006.

[49] Conallen J. Modeling Web application architectures with UML. *Communications of the ACM*, 42(10):63–70, 1999.

[50] Bodoff S.; Armstrong E.; Ball J.; Bode D.; Evans I.; Green D.; Haase K.; Jendrock E. *The J2EE Tutorial*, pages 2–10. Addison Wesley, 2004.

[51] Deploying PlanetLab: An Open Platform for Developing and Accessing Planeatary-Scale Services. http://www.planet-lab.org.

[52] Geni: Global Environment for Network Innovations. http://www.geni.net.