MODELING AND SIMULATION OF POINT SPREAD FUNCTIONS FOR ADVANCED SAR SYSTEMS

By

Hilaura Raquel Nava Valles

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

MASTER OF SCIENCE
in
ELECTRICAL ENGINEERING
(Digital Signal Processing)

University of Puerto Rico
Mayagüez Campus
2004

Approved by:

Shawn Hunt, Ph.D. Member, Graduate Committee	Date
Manuel Jiménez, Ph.D. Member, Graduate Committee	Date
Domingo Rodríguez, Ph.D. President, Graduate Committee	Date
Barbara Calcagno, M.S. Representative of Graduate Studies	Date
Jorge Ortiz, Ph.D. Chairperson of the Department	Date

ABSTRACT

MODELING AND SIMULATION OF POINT SPREAD FUNCTIONS FOR ADVANCED SAR SYSTEMS

By

Hilaura Raquel Nava Valles

This work deals with the treatment of remotely sensed or sensory data in order to extract important information to the users. Sensory data are data collected from physical systems, by using sensors, such as the synthetic aperture radar (SAR) system. A fundamental stage in SAR processing is computation of its point spread function (PSF), necessary for raw data generation. The PSF contains the backscattered energy of a single point target on the Earth, allowing the extraction of surface information. The effort of this work is concentrated in the development of a mathematical model for PSF simulations and SAR raw data generation. The SAR impulse response is modeled in a time-frequency analysis context as the radar ambiguity function of a single point in the spatial object domain. The simulation algorithms were programmed using the functional programming style to get modularity. These algorithms were developed and tested in Matlab. The mathematical model was validated using a methodology based on theoretical formulations available in the literature. Simulations results demonstrated that not only the model is valid, but also the efficiency of the implemented algorithms as a tool for modeling and simulation of SAR impulse response functions.

RESUMEN

MODELAJE Y SIMULACION DE RESPUESTA DE IMPULSO PARA SISTEMAS SAR AVANZADOS

Por

Hilaura Raquel Nava Valles

Este trabajo atiende el tratamiento de datos obtenidos remotamente a través de sensores o datos sensoriales, con la finalidad de extraer información importante para los usuarios de dichos datos. Estos datos son colectados por sistemas físicos de percepción remota, tales como, los sistemas de radar de abertura sintética (SAR). Una etapa fundamental en el procesamiento de datos de SAR es la computación de la función de extensión de punto (PSF, por sus siglas en inglés) para la generación de los datos crudos, ya que ésta contiene la energía reflejada de un sólo punto sobre la Tierra, permitiendo la extracción de información de la superficie. El esfuerzo de este trabajo se concentró en el desarrollo de un modelo matemático para la simulación de la respuesta de impulso y la generación de datos crudos de los sistemas SAR. La respuesta de impulso de los sistemas SAR fue modelada en un contexto de análisis de tiempo-frequencia como la función de ambigüedad de un sólo punto en el dominio espacial del objeto. Los algoritmos para las simulaciones fueron desarrollados y probados en Matlab. El modelo matemático fue validado con una metodología basada en formulaciones matemáticas disponibles en la literatura. Los resultados obtenidos de las simulaciones no solo demostraron que el modelo es válido, sino también la eficiencia de los algoritmos implementados como una herramienta para el modelaje y simulación de la respuesta de impulso de los sistemas SAR.

To the people who will never leave me alone: God, my loved family, and my unconditional friends.

ACKNOWLEDGMENTS

I would like to thank deeply my thesis advisor, Dr. Domingo Rodríguez for his valuable encouragements and advises during the preparation of this work. Thank you to my graduate committee members, Dr. Shawn Hunt, and Dr. Manuel Jiménez, for their help.

I would like to thank the Electrical and Computer Engineering department, and the PRECISE Project for their financial support and the research facilities provided during these years of study. Also, i would like to convey my thanks to my friends of the PRECISE laboratory for all their support.

TABLE OF CONTENTS

LIST OF TABLES ix

LIST OF FIGURES x

1 Introduction 1

1.1 Problem Statement . . . 3
1.2 Justification . . . 4
1.3 Objectives . . . 5
1.4 Thesis Overview . . . 5

2 Previous Research Work 7

2.1 Previous Research Work on Ambiguity Function Processing . . . 7
2.2 Previous Research Work on SAR Processing Modeling and Simulations . . . 10

3 Mathematical Framework 15

3.1 Basic Linear Algebra Definitions . . . 15
3.2 Finite Dimensional Signal Algebra . . . 23
3.3 Signal Algebra Operators on $\ell^2(Z_N)$ . . . 26
3.3.1 Unary Signal Operators . . . 27
3.3.2 Binary Signal Operators . . . 27
3.3.3 Matrix Representation of Signal Operators . . . 29
3.3.3.1 Right Shift Operator $S_N^{(-)}$ . . . 29
3.3.3.2 Left Shift Operator $S_N^{(+)}$ . . . 31

4 Modeling of Point Spread Functions as Discrete Ambiguity Functions 33

4.1 Introduction to Modeling and Simulation . . . 34
4.1.1 Modeling . . . 34
4.1.2 Simulation . . . 35
4.2 Radar Imaging Systems . . . 36
4.2.1 Synthetic Aperture Radar (SAR) . . . 40
4.2.1.1 Mathematical Model of Radar Imaging Systems . . . 42
4.2.2 SAR Imaging System Model . . . 43

4.3 The Radar Ambiguity Function 46
4.3.1 Ambiguity Function of Basic Waveforms 49
4.3.2 Discrete Ambiguity Function (DAF) 51
4.4 Algorithms for DAF Computation 53
4.4.1 Analysis of Complexity 57

5 Simulation Results **58**
5.1 Results on Modeling 58
5.2 MATLAB Simulations Results 59
5.2.1 Graphical Results of MATLAB Simulations 61
5.2.2 Comparison between Linear and Cyclic DFT Methods 65
5.3 Methodology Validation 67
5.4 Future Work 71

6 Conclusions **75**

BIBLIOGRAPHY **76**

A Programming with FFTW **79**

B MATLAB Functions **86**

C C Programs **100**

D FORTRAN Programs **107**

LIST OF TABLES

4.1 Radar Operational Frequency Bands 37
5.1 Time Measurements of Matlab Simulations 60
5.2 Simulation Parameters for DFT Method 65
5.3 Time Measurements of SAR PSFs Simulations on a SMP System 73
5.4 Time Measurements of SAR PSFs Simulations on the IBM Cluster 74

LIST OF FIGURES

1.1 A Typical Imaging Radar System 2
1.2 Filtering View of the Imaging Process. 4
2.1 SAR Processing Oriented Simulator 11
2.2 SAR Oriented Simulator 12
2.3 Strimap SAR Operation Mode. 13
2.4 Scan SAR Operation Mode. 13
2.5 Spotlight SAR Operation Mode. 14
3.1 Algebraic Structures 22
3.2 Unary Signal Algebra Operator 27
4.1 The Simulation Process 36
4.2 Resolution Cell of a Radar Imaging System 38
4.3 Range Resolution 38
4.4 Azimuth Resolution 39
4.5 Doppler Variation Computation 41
4.6 Blahut Model of Imaging Radar Systems 43
4.7 SAR System Model 44
4.8 SAR Raw Data Generation Model 45
4.9 SAR Image Formation Model 46
4.10 The Ambiguity Function of a Single Pulse ($\tau = 4$). 3-D plot (left). Contour plot (right) 50
4.11 The Ambiguity Pulse of 7 Coherent Pulses ($\tau = 0.4$ and $PRI = 1$).3-D plot(left). Contour plot (right) 51
4.12 Discrete Ambiguity Function as a Cyclic Correlation 52
4.13 Index-Reversal and Zero-Padding Operators 53
4.14 DFT Method Algorithm for Ambiguity Function Simulations 54
4.15 Filter Method Algorithm for Ambiguity Function Simulations 56
5.1 SAR imaging system 58
5.2 Three-dimensional Representation of the Ambiguity Function for a Transmitted Pulse Signal 62
5.3 Two-dimensional Representation of the Ambiguity Function for a Transmitted Pulse Signal 62
5.4 Contour Plot of the Ambiguity Function for a Transmitted Pulse Signal . . 63

5.5 Three-dimensional Representation of the Ambiguity Function for a Transmitted Chirp Signal . . . 63
5.6 Two-dimensional Representation of the Ambiguity Function for a Transmitted Chirp Signal . . . 64
5.7 Contour Plot of the Ambiguity Function for a Transmitted Chirp Signal . . 64
5.8 Chirp Ambiguity Function Computed by the Linear DFT Method 66
5.9 A Chirp Ambiguity Function Computed by the Cyclic DFT Method 66
5.10 Validation Methodology for PSF simulations 67
5.11 Validation Methodology for PSF simulations for a Transmitted Pulse Signal 67
5.12 Single Pulse Ambiguity Function Computed directly from Equation 68
5.13 Single Pulse Ambiguity Function Computed with the DFT Method 69
5.14 Validation Methodology for PSF simulations for a Transmitted Pulse Train 69
5.15 Pulse Train Ambiguity Function Computed directly from Equation 70
5.16 Pulse Train Ambiguity Function Computed with Filter Method 70
5.17 SAR Typical Data Size 71
5.18 Parallel Approach for SAR Impulse Response Simulations 72
5.19 Run Time Comparison of SMP Simulations 74
A.1 FFTW Computational Scheme 81

CHAPTER 1

Introduction

The continuous monitoring of the Earth's surface by remote sensing systems has contributed with a diversity of geoscience applications in many fields, such as: geology, hydrology, oceanography, and others. As new technologies and innovative concepts are developed, the number of applications increase. For instance, the last decade has witnessed significant advances in our understanding of global geophysical phenomena as a direct consequence of the development of *radar imaging technology* for sensor systems design.

Imaging radars are systems designed to operate at low intensity in the microwave region of the electromagnetic spectrum. Basically, the operation of a radar imaging system starts with the transmission of a sequence of short pulses at a specific wavelength and polarization while the system platform flies along its trajectory [1]. Then, all returned signals containing Earth's surface reflectivity information are combined to form an image known as raw data. Figure 1.1 depicts a classic radar imaging system and its fundamental elements. Depending of the antenna length used for signals transmission and reception, remote sensing radars can be classified into two categories: real aperture radars (RAR) and synthetic aperture radars (SAR). The main difference between these systems reside in their spatial resolution capability. The spatial resolution of any radar imaging system can be defined as the system's ability to resolve smallest distance between two or more point targets that are sufficiently separated so that allow individual data measurements among

them. This important feature of the radar imaging systems is characterized by its point spread function (PSF) or impulse response function.

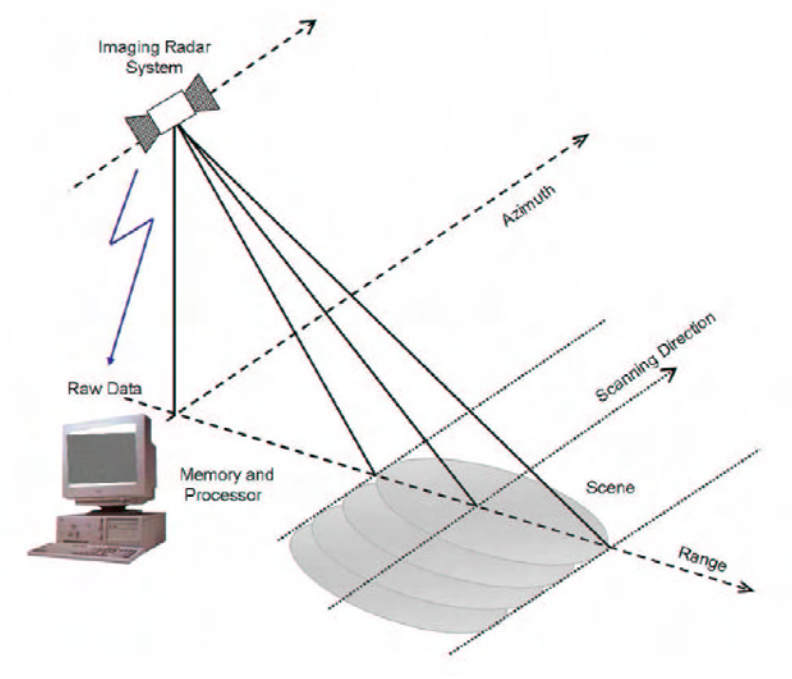

Figure 1.1: A Typical Imaging Radar System

This work, deals with the modeling and simulation of SAR systems impulse response function. For this, is necessary to concentrate on the formulation of a model for SAR raw data generation that be used in the study of the characteristics of SAR impulse response function. Finite dimensional signal algebra operators was employed for the formulation of the SAR point spread function model.

Modeling is defined in this research as a set of mathematical structures and equations designed to correspond to a physical system or entity based on a set of prescribed assumptions [2]. A system, in general, is defined as a set of elements and its interrelationships; whereas, simulation is defined as the execution of a modeling system through computer programming [2]. The computational effort to process SAR data is expensive in

terms of time and hardware requirements due to the large amounts of sensory data collected by the system. In this regard, is of interest the formulation of an adequate structure to perform SAR processing simulations. Such structure is known as automated information system (AIS). An AIS is defined here as a combination of computer hardware and software; configured to accomplish specific information-handling operations, such as communication, dissemination, processing, and information storage [3].

The specific application problem of interest in this work is wetlands monitoring using radar imaging systems to estimate Earth's surface moisture content. Wetlands are considered as one of the most productive and biologically diverse environments in the world. According to the national Wetlands Inventory, the productivity of a wetland is measured by the total weight of plant and animal material produced per unit area [4]. In this sense, about 6.4 percent of the planet is covered by wetlands, however they account for 24 percent of total global productivity [4]. Recent estimates show that the world may have lost 50 percent of the wetlands that existed since 1900. For example, in the USA, more than 50,000 acres of wetlands are lost annually due to different causes, such as, draining and filling of wetland, and chemical contamination. This work will contribute to the surface image analysis effort been conducted in such geophysics applications.

1.1 Problem Statement

The main problem adressed by this thesis work can be stated in the following manner: There is a need to develop a discrete-time, discrete-frequency computational model to characterize the imaging kernel in a SAR imaging system. The imaging kernel is defined as the weight function in a convolution operation with the reflectivity density function which represents the imaging process. To characterize the imaging kernel of a SAR system implies to describe all its attributes pertaining to signal processing operations. R. Blahut [5] suggests in his work on remote surveillance algorithms to use the ambiguity function as

the weight function or imaging kernel of radar imaging systems . The radar imaging system model conceived by Blahut is a continuous model where the system expected output can be viewed as a two-dimensional convolution operation of the point reflectivity density function with the radar ambiguity function, as shown in Figure 1.2.

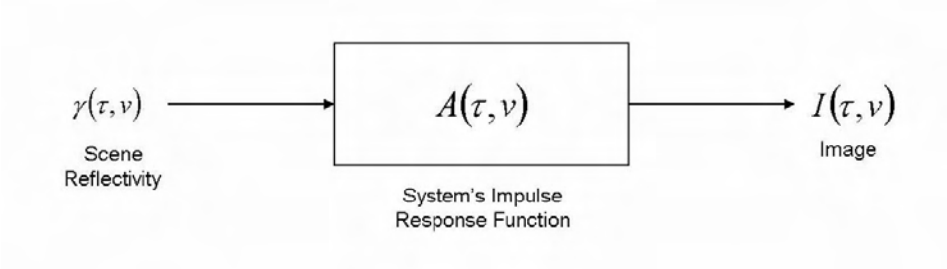

Figure 1.2: Filtering View of the Imaging Process.

1.2 Justification

At the present time there is a great demand at the academic level for low-cost discrete-time discrete-frequency modeling and simulation tools that could aid in the further understanding of the imaging process of SAR systems. Having discrete-time discrete-frequency models will facilitate to a great extent the design of an entire "all digital" SAR imaging processing system. It will also facilitate the use of advanced digital signal processing techniques in the hardware/software implementations of such systems, taking into consideration issues such as modularity, scalability, and reconfigurability.

The work presented in this thesis adressed the problem of discrete-time discrete-frequency modeling for SAR imaging systems by formulating a new mathematical model utilizing an operator theoretic approach over a signal algebra. The signal algebra was constructed using the cyclic or circular convolution operation as the vector product operation in a finite dimensional complex vector space. The operator theoretic approach taken in this work allowed the introduction af the concept of modular design in the algorithms developed

to simulate the SAR imaging model. A relationship was established between the modular design concept and the functional programing style used in algorithm source coding. The author considered this established relationship a major contribution.

1.3 Objectives

Main Objective

The major goal behind this work is to create models and a simulation environment that allows the study and analysis of impulse response functions characteristics of advanced SAR systems.

In order to achieve the above enunciated goal, the following secondary objectives were accomplished:

- Understand the SAR concept, which involves processes such as: data acquisition, impulse response function computation, raw data generation and image formation process.
- Formulate a mathematical model for SAR raw data generation simulation.
- Formulate a mathematical model for SAR image formation simulation.
- Perform point spread functions simulation by means of efficient radar ambiguity function processing.

1.4 Thesis Overview

In chapter two, *Previous Research Work*, a summary of the most relevant previous works associated to ambiguity function and SAR processing is provided. Chapter three, *Mathematical Framework*, introduces some mathematical preliminaries concepts, and theory about linear algebra and vector spaces. Chapter four, *Modeling of Point Spread Functions*

as Discrete Ambiguity Functions, presents in detail the mathematical formulations for the modeling of SAR point spread functions as discrete ambiguity functions. The fifth chapter, *Simulation Results*, presents computational implementations and results of the implementation of the algorithms for PSF simulations. Finally, chapter seven, *Research Conclusions*, presents the thesis conclusions.

CHAPTER 2

Previous Research Work

In the last decade, more applications areas have been developed for the imaging radar research, such as: remote sensing, Earth surface surveillance and automatic target recognition (ATR) applications. In this sense, the production of a data set of high quality, be it in analog or digital format, is required for the accurate interpretation and analysis. In the particular case of SAR processing, many efforts have been conducted through the years in order to improve the treatment of the collected data and thus the images resolution quality. Some of the most significant works that have contributed to SAR processing and those related with the radar ambiguity function processing are summarized below.

2.1 Previous Research Work on Ambiguity Function Processing

Most of the research work conducted in the ambiguity functions processing area has been focused on theoretical representations and analysis. The integration of theory, algorithms formulation, and computer implementations is poorly evidenced in the current available literature. In this regard, a brief chronological explanation of certain relevant works on radar ambiguity functions processing is presented below, emphasizing its main contributions to this work.

In 1984, Auslander and Tolimieri, developed a theory for the characterization of radar ambiguity functions in terms of the cross-ambiguity functions associated with a particular orthonormal basis of signal space resulting from the rectangular pulse [6]. The set of ambiguity functions associated to the orthonormal standard basis is presented here as another orthonormal basis used to write the ambiguity function of any signal. The theorems proposed in this work for the characterization of ambiguity functions are applied to stablish two important mathematical results: the set of ambiguity functions is closed on the square-integrable topology, and the sum of two ambiguity functions is never an ambiguity function.

A year later, 1985, Tolimieri and Winograd, presented for the first time an algorithmic formulation for the computation of the discrete ambiguity function [7] using signal processing operators. The proposed algorithms are based on two different approaches depending on how the ambiguity function be written. These approaches gave rise to the development of the following methods for computing the discrete ambiguity function: the filter method and the transform method. The filter method consists of computing the ambiguity function as a filter or cyclic convolution operation. The formulated algorithm applies the object-domain convolution theorem allowing cyclic convolution be computed in an efficient manner through indirect methods using fast Fourier transform (FFT) algorithms. In the transform method the ambiguity function is computed directly by means of the one-dimensional discrete Fourier transform (DFT). The computational complexity of both algorithms in terms of complex multiplications operations is presented in this work. In this sense, the transform method resulted more computationally efficient than the filter method.

In 1988, Auslander and Tolimieri, exploited the parallelism intrinsic in the computation of the discrete ambiguity function [8] through a computational method called the transform method [7]. This method was redesigned to compute the ambiguity surface

by means of the two-dimensional Fourier transform as the major step of this algorithm. Only decimated values of the ambiguity function, ranging over selected points in the entire domain, can be computed with the algorithm proposed in this work [8]. The authors, presented a novel way to implement the two-dimensional FFT operation on a parallel architecture by means of computing multiple one-dimensional FFTs. However, results on computer implementations of the algorithm were not presented.

In 1993, Rodríguez, Seguel and Cruz formulated a methodology based on algebraic methods for the analysis, design, and modifications of time-frequency signal processing algorithms [9]. These algebraic methods are integrated within a computational mathematics environment (CME) developed to assist the implementation of time-frequency algorithms on a given computational hardware structure (CHS). This environment allows the design of algorithms as a composition of several signal processing operators that can be described using computational mathematics structures (CMS), such as, the tensor product algebra. The filter method algorithm was formulated in this work in terms of linear, finite signal operators acting on finite complex sequences [7]. Also, a variant of this algorithm was obtained using algebraic methods and some properties of the operators.

Richman, Parks and Shenoy, presented in 1998 an extension of their work on discrete time-frequency representations introduced in 1995 [10]. Group representation theory is used in this work for the formulation of discrete-time, discrete-frequency Wigner distribution for analysis of discrete-time, periodic signals [11]. The ambiguity function is related to the Wigner distribution through the Fourier transform, i.e., obtaining Wigner distribution is possible taking the two-dimensional Fourier transform of the discrete ambiguity function. An important contribution of this work is the formulation of time-frequency representations in terms of the Heisenberg group, which is the group corresponding to discrete cyclic shifts.

For year 2001, Özdemir and Arikan proposed new algorithms to compute uniformly spaced samples of the ambiguity function and the Wigner distribution on arbitrary line segments [12]. This approach is based on the fractional-Fourier transformation (FrFT) of the time-domain signals to obtain novel closed-form expressions for the slices of the ambiguity function and the Wigner distribution. Algorithm implementation for the ambiguity function computation uses a digital computation algorithm for the FrFT proposed by H. M. Ozaktas that compute the required samples in O(NlogN) flops [13] .

In 2002, Mozeson and Levanon presented a publicly available Matlab code for academic and pedagogical purposes [14] . The code computes and plots the ambiguity function of many different radar signals. The program computes only two of the four quadrants of the ambiguity function because the symmetry of this function respect to the origin.

2.2 Previous Research Work on SAR Processing Modeling and Simulations

In this section, a review of some research works related to SAR processing modeling and simulations is presented. Special attention is given to those works associated to SAR raw data generation.

G. Franceschetti, M. Migliaccio and D. Riccio proposed two different approaches for SAR processing simulation: the SAR processing oriented simulation and the SAR oriented simulation [15]. The SAR simulator is based on the generation of the raw signal starting from the reflectivity function (right side figure 2.1) as the counter part of the SAR processor (left side) that perform an inversion process to retrieve the scene reflectivity. This approach is termed as processing oriented simulator because it is based on the inversion of the SAR processor operational mode. On the other hand, the SAR oriented simulator depicted in figure 2.2, includes models of the electromagnetic interactions of the SAR signal with the

scene in order to estimate some geophysical parameters from the reflectivity. However, the upper blocks in figure 2.2 are difficult to implement due to the non-linear effects (foreshortening, layover and shadowing) created by the geometric non-linear projection that SAR operates from the ground to the slant coordinate system. Therefore, this block is not being well defined, and most important is that the inversion procedure (right side) not aimed at producing the same parameters as shown at the input of the chain (left side).

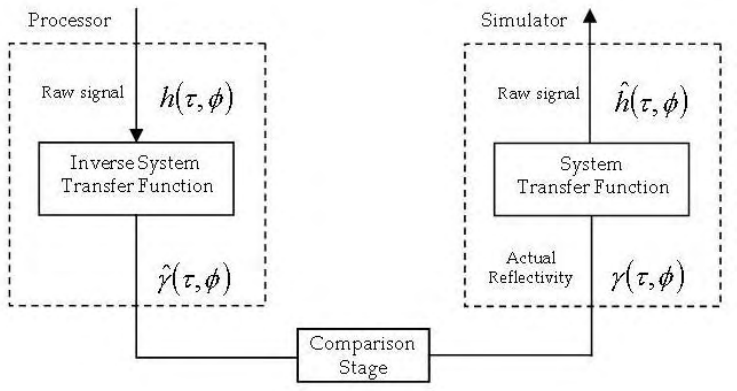

Figure 2.1: SAR Processing Oriented Simulator

There are three main operating modes of SAR systems: stripmap, scan, and spotlight [16]. In stripmap mode, the radar antenna pointing is fixed in a direction with respect to the flight platform path as in figure 2.3. The antenna footprint covers a strip on the illuminated surface as the platform moves and the system operates. The image generated is limited in the range direction but not in the azimuth.

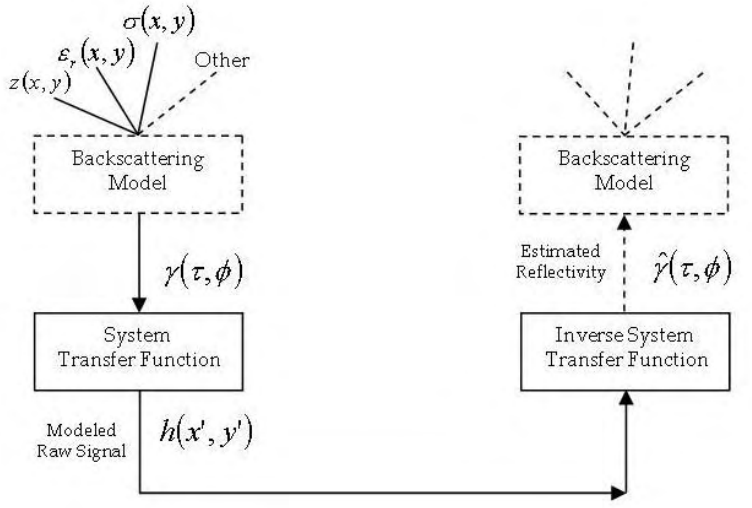

Figure 2.2: SAR Oriented Simulator

In scan mode the antenna beam is periodically stepping to neighboring subswaths in the range direction (see figure 2.4) allowing an increase of the range swath dimension. Finally, in spotlight mode the radar steers its beam during the overall acquisition time to illuminate the same area as is presented in figure 2.5. It allows an increment of the available antenna length improving azimuth resolution.

G. Verdone et al. presented a description of the existing processing algorithms implemented to focus images obtained in the three different operational modes of SAR systems [17]. For the scanSAR mode a time domain algorithm is presented which consists of perform range and azimuth compression separately

For the spotlight operational mode of SAR systems, a raw data simulator of extended scenes was proposed by G. Franceschetti, A. Iodice, D. Riccio and G. Ruello [18]. This approach is based on the two-dimensional Fourier transformation. The proposed simulator is an extension of the efficient frequency domain approach presented by the same authors for the stripmap mode [19]. This simulator consists of two main stages: the first one is the reflectivity map evaluation given the orbit data and the scene geometric and

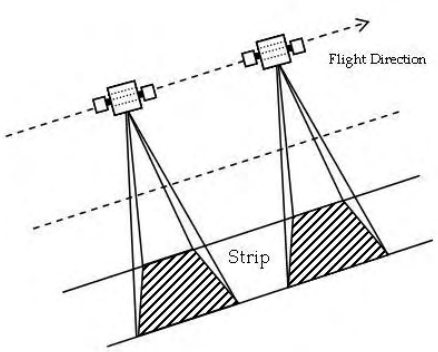

Figure 2.3: Strimap SAR Operation Mode.

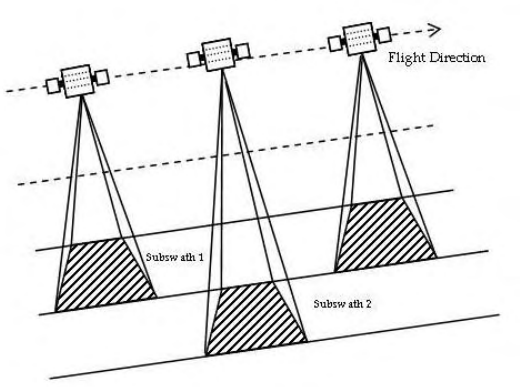

Figure 2.4: Scan SAR Operation Mode.

electromagnetic parameters. The second one corresponds to the SAR raw signal computation by weighting the reflectivity map by the SAR system two-dimensional PSF. For the spotlight mode case, the system transfer function depends on the azimuth coordinate of the surface point, and then the spectral domain formulation is not straightforward. However, this problem is overcome by considering a long acquisition time, and then truncating the obtained raw signal. This approach meets some stringent requirements such as: be able to deal with extended scenes, and not only with a limited number of point scatterers. The simulating algorithm is efficient and time and memory saving.

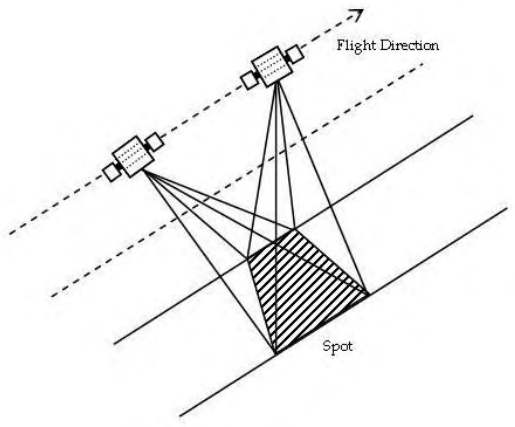

Figure 2.5: Spotlight SAR Operation Mode.

CHAPTER 3

Mathematical Framework

The aim of this chapter is to present a mathematical framework to be utilized for the theoretical formulation of modeling and simulation of the impulse response function of synthetic aperture radar (SAR) systems. The main objective of the theoretical formulation is to model SAR imaging systems as a discrete two-dimensional space invariant linear systems. In addition, this theoretical framework was used to facilitate the transition from mathematics to algorithmic formulations in a systematic and comprehensible manner.

3.1 Basic Linear Algebra Definitions

In this section, some fundamental notions on linear algebra are briefly recalled, such as, basic algebraic structures, vector spaces, and linear operators. A more extensive introduction to these subjects can be found in [20] and [21]. These definitions are used to formulate a finite dimensional signal algebra. This section is useful to understand the main results of this research, and the notation used in later chapters.

Definition 3.1 *A **set** is any collection of objects such that these objects are members of the set. If there is a positive integer n such that a set, say A, contains exactly n, and not more different elements, the set A is finite; otherwise A is infinite [22].*

For instance, some important sets of numbers are introduced, such as:

1. $\mathbb{Z} = \{\ldots -2, -1,\ 0,\ \ 1,\ \ 2, \ldots\}$ *which corresponds to the set of all integer numbers.*
2. $\mathbb{R} = \{t : t \text{ is a real number}\}$*, the set of all real numbers.*
3. $\mathbb{C} = \left\{\alpha = a_r + jb_i : a_r, b_i \in \mathbb{R};\ j = \sqrt{-1}\right\}$ *which correspond to the set of all complex numbers.*

Definition 3.2 *Let A and B be two arbitrary sets of elements. If every element of A also belongs to B, then A is called a* ***subset*** *of B [23]. For example, the set of integer numbers $\mathbb{Z}$ is a subset of the set of real numbers $\mathbb{R}$, and this is a subset of the set of complex numbers $\mathbb{C}$.*

Definition 3.3 *For two arbitrary sets of elements, say A and B, denoted by $A = \{a_k : a_k \in A\}$ and $B = \{b_l : b_l \in B\}$; the* ***Cartesian product*** *of A and B is defined as a new set, denoted by $C = A \times B$. This cartesian product is formed in the following manner:*

$$C = \{(a_k, b_l) : a_k \in A\ ,\ b_l \in B\} \tag{3.1}$$

The Cartesian product of two sets of elements is not commutative, i. e., $A \times B \neq B \times A$

Definition 3.4 *A* ***relation*** *$\rho \subset A \times B$, of a cartesian product $A \times B$ is any subset of the cartesian set. In this case, A is related to B through the relation ρ. For a relation $\rho \subset A \times B$, the following notation is presented:*

$$\rho : \begin{array}{ccc} A & \rightarrow & B \\ a_k & \mapsto & b_\ell \end{array};\ (a_k, b_\ell) \in \rho;\ b_\ell = \rho\,(a_k) \tag{3.2}$$

$\rho : A \rightarrow B$ is used to denote a relation ρ from a set A to a set B.

Definition 3.5 *A* ***function****, say $f \subset A \times B$, is any relation in a cartesian set $A \times B$, where each first entry of every ordered pair appears once and only once.*

Definition 3.6 *A **mapping** is any function, say $f : A \to B$. The domain of the function is the entire set A and the co-domain of the function is the entire set B. The range of the function is the set $r(f) = \{b_\ell : a_k \text{ and } f(a_k) = b_\ell\}$*

Definition 3.7 *A **numeric function** say $g \subset A \times B$, is a relation where, both, A and B are sets of numbers; for example the set of complex numbers or the set of integer numbers. The first set, A, is called the domain of the function, and the second set B is called the co-domain. The following notation is used in this work for a numeric function.*

$$\begin{aligned} g: \quad & A \to B \\ & a_k \longmapsto g(a_k) = b_l \text{ or } (a_k, b_l) \in g \subset A \times B \end{aligned} \tag{3.3}$$

A function is complex or is real depending on its co-domain. If its co-domain is the set of real numbers $\mathbb{R}$, the function is real. If the function's co-domain is the set of complex numbers $\mathbb{C}$, then the function is complex.

Definition 3.8 *A function, say x, defined by its domain, which is always a discrete set or countable is called a discrete function. In this work, we call **discrete signal** to a discrete function.*

Example 3.1 *Discrete cosine function*

The set of integers $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is a discrete set. Then, it is had a discrete cosine signal which domain is the set $\mathbb{Z}$ as follows:

$$\begin{aligned} x: \quad & \mathbb{Z} \to \mathbb{R} \\ & n \longmapsto x[n] = \cos 2\pi f_0 n T_s \end{aligned} \tag{3.4}$$

Square brackets are used to enclose the domain valuations of a discrete signal. An index set is defined here by any set used as domain for a discrete signal. The **standard finite indexing set** will be used.It is a finite subset of the set of integers given by: $\mathbb{Z}_N = \{0, 1, 2, \ldots N-1\}$.

Definition 3.9 *A **digital signal** is defined by its co-domain, which is always a finite, discrete set.*

Example 3.2 *Continue-digital signal*

$$\mu: \begin{array}{l} \mathbb{R} \to \{0,1\} \\ t \longmapsto \mu(t) \end{array}, \text{ where } \mu(t) = \begin{cases} 1 & , \quad t \geqslant 0 \\ 0 & , \quad t < 0 \end{cases} \tag{3.5}$$

Example 3.3 *Discrete-digital signal*

$$y: \begin{array}{l} \mathbb{Z} \to \mathbb{Z}_N \\ k \longmapsto y[k] = \langle k \rangle_N \end{array}, \text{ where } \langle k \rangle_N = remainder\left(\frac{k}{N}\right) \tag{3.6}$$

Definition 3.10 *A **binary operation** takes any two arbitrary elements from a particular set, say A, and it relates them to a new element that also belongs to the set A. In other words, a binary operation on a set A is a mapping of the form $f: A \times A \to A$. The binary operation of two arbitrary elements a and b of a set A is denoted as $f(a,b)$.*

Definition 3.11 *Let G be an arbitrary set that contains n elements and a single binary operation defined between any two elements of G . This binary operation can be an arithmetic operation such that addition is denoted by $(+)$, or multiplication is denoted by $(\cdot)$. The set G is called a **group** if it satisfies the following fundamental properties [24]:*

- Closure: *If a, b are elements of G, then the multiplication operation denoted by $a \cdot b$ results in an element of G.*
- Identity: *There exists an element $e \in G$ such that $a \cdot e = a$, for all $a \in G$.*
- Inverse: *For every $a \in G$, there is an element $b \in G$ such that $a \cdot b = e$.*
- Associativity: *For arbitrarily chosen elements of G we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.*

Definition 3.12 *A commutative group or an **Abelian group** is a group that satisfies the commutative property under its binary operation. This property establishes that for every*

element $a, b \in G$ *we have* $a \cdot b = b \cdot a$.

Definition 3.13 *A* ***ring*** *is any set, say* R*, with two associated binary operations (addition and multiplication) satisfying the following conditions:*

- Additive Identity: *For every* $a \in R$*, there is an element* $0 \in R$ *such that* $a + 0 = a$.
- Inverse: *For every* $a \in R$*, there is an element* $-a \in R$ *such that* $a + (-a) = 0$.
- Additive Associativity: *For every* $a, b, c \in R$ *we have* $(a + b) + c = a + (b + c)$.
- Additive Commutativity: *For every* $a, b \in R$ *we have* $a + b = b + a$.
- Multiplicative Associativity: *For every* $a, b, c \in R$ *we have* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- Distributivity: *For every* $a, b, c \in R$ *we have* $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

Thus, a ring is an Abelian group under the addition operation. A ring that satisfies the commutative property under the multiplication operation is called a commutative ring.

Definition 3.14 *An* ***integral domain****, say* D*, is a commutative ring that has an identity element. Also, an integral domain satisfies the cancellation property under the multiplication operation. This property establishes that the product of every two non-zero elements, say* $a, b \in D$*, is always a non-zero element [25]. The set of integer numbers* $\mathbb{Z}$ *forms an integral domain.*

Definition 3.15 *A* ***field*** F *is defined as a commutative ring for which the associative, commutative, and distributive properties hold under its binary operations. This also contains both, a multiplicative and an additive identity element, as well as, a multiplicative and an additive inverse element [24]. In other words, a field is a special set of elements, which can be added and multiplied under defined conditions.*

For instance, the sets of complex and real numbers, $\mathbb{C}$ *and* $\mathbb{R}$ *respectively, satisfy all the conditions to be fields.*

Definition 3.16 *A **linear vector space**, ℓ, defined over a specific field, say F, is a set composed of elements called vectors that has two associated arithmetic operations such as addition and scalar multiplication [26]. If the field over which the vector space is defined is an infinite set, such as the set of complex numbers $\mathbb{C}$, then we have an infinite linear vector space. Any linear space must satisfy the following conditions:*

1. *Homogeneity:*

   For a vector space ℓ, and any $v_0 \in \ell$, and any scalar $a \in F$, then it is had $a \cdot v_0 \in \ell$

2. *Superposition:*

   For a vector space ℓ, and any $v_0, v_1 \in \ell$, we have $v_0 + v_1 \in \ell$

Definition 3.17 *Let F^n be the **set of n-tuples elements** of the field F. Let B and C be two arbitrary vectors of F^n, denoted by $B = [b_0, b_1, b_2, \ldots, b_{n-1}]$ and $C = [c_0, c_1, c_2, \ldots, c_{n-1}]$ respectively. Then, the addition of B and C is given by:*

$$B + C = [b_0 + c_0, b_1 + c_1, b_2 + c_2, \ldots, b_{n-1} + c_{n-1}] \tag{3.7}$$

Let γ be a scalar that belongs to F, then the product of γ and B is defined by:

$$\gamma \cdot B = [\gamma \cdot b_0, \gamma \cdot b_1, \gamma \cdot b_2, \ldots, \gamma \cdot b_{n-1}] \tag{3.8}$$

Definition 3.18 *Let ℓ be a vector space over the field F, and $v_0, \ldots, v_{n-1}$ are vectors in ℓ. Then a vector $\bar{v} \in \ell$ is called a **linear combination** of the vectors v_i if it can be written as follows:*

$$\begin{aligned} \bar{v} &= a_0 \cdot v_0 + a_1 \cdot v_1 + \ldots + a_{n-1} \cdot v_{n-1} \\ &= \sum_{i=0}^{n-1} a_i \cdot v_i \quad , \quad a_i \in F \end{aligned} \tag{3.9}$$

Definition 3.19 *Let $v_0, v_1, \ldots, v_{n-1}$ be elements of a vector space ℓ over a field F. The vectors $v_0, v_1, \ldots, v_{n-1}$ are **linearly independent**, if and only if, the following condition is satisfied on its linear combination:*

$$a_0 \cdot v_0 + a_1 \cdot v_1 + \ldots + a_{n-1} \cdot v_{n-1} = O \tag{3.10}$$

Then, $a_i = 0$ for all $i = 0, 1, 2, \ldots, n-1$. Notice that the symbol O on the right is the zero element in the vector space ℓ, not the zero element in the field F.

Definition 3.20 *Let $\{v_0, v_1, \ldots, v_{n-1}\}$ be a set of vectors linearly independent that belongs to the vector space ℓ. Then, it can be said that the set $\{v_0, v_1, \ldots, v_{n-1}\}$ is a **basis** of ℓ.*

Definition 3.21 *The **inner product** between two arbitrary vectors, A and B, is defined as the dot product between these vectors as follows: $A \cdot B = a_0 \cdot b_0 + a_1 \cdot b_1 + \ldots + a_{n-1} \cdot b_{n-1}$. This operation produces a scalar, for this reason, is also known as scalar product. The inner product between two vectors is shortly denoted by $\langle A, B \rangle$.*

Definition 3.22 *Let $\mathbb{Z}_N$ be a finite subset of integer numbers given by $\mathbb{Z}_N = \{0,\ 1,\ 2, \ldots, N-1\}$. Any vector space conformed by a basis with a finite number of elements, is called **finite dimensional vector space** and is denoted by $\ell(\mathbb{Z}_N)$.*

Definition 3.23 *A **Hilbert space** is a linear finite-dimensional vector space that has an associated inner product. This vector space is defined over the field of complex numbers $\mathbb{C}$ and is denoted by $\ell^2(\mathbb{Z}_N)$.*

Definition 3.24 *A **linear algebra** over a field F, is a vector space ℓ which has a defined binary operation, such as, the vector multiplication. This operation relates two vectors, $v_0, v_1 \in \ell$ to a new vector $v_0 \cdot v_1 \in \ell$ called the product of v_0 and v_1, such that, the following conditions are satisfied:*

- *The multiplication is associative: $v_0 \cdot (v_1 \cdot v_2) = (v_0 \cdot v_1) \cdot v_2$*

- *The multiplication is distributive with respect to the addition:* $v_0 \cdot (v_1 + v_2) = v_0 \cdot v_1 + v_0 \cdot v_2$ *and* $(v_0 + v_1) \cdot v_2 = v_0 \cdot v_2 + v_1 \cdot v_2$
- *For every scalar* $a \in F$, $a\,(v_0 \cdot v_1) = (a \cdot v_0) \cdot v_1 = v_0 \cdot (a \cdot v_1)$

If there exists an element 1 *in* ℓ*, such that,* $1 \cdot v_0 = v_0 \cdot 1 = v_0$ *for every vector* $v_0 \in \ell$*,* ℓ *becomes a linear algebra with unity over* F*, and* 1 *is called the identity of* ℓ*. This algebra is said commutative if* $v_0 \cdot v_1 = v_1 \cdot v_0$ *for every* $v_0, v_1 \in \ell$ *[20].*

Definition 3.25 *An* ***algebraic structure*** *is an arbitrary set with one or more operations defined in the set. While more operations are defined on the set, richer in structures is considered to be the set.*

The algebraic structures introduced in previous definitions are illustrated in figure 3.1. Notice that the algebra is the richest structure in operations.

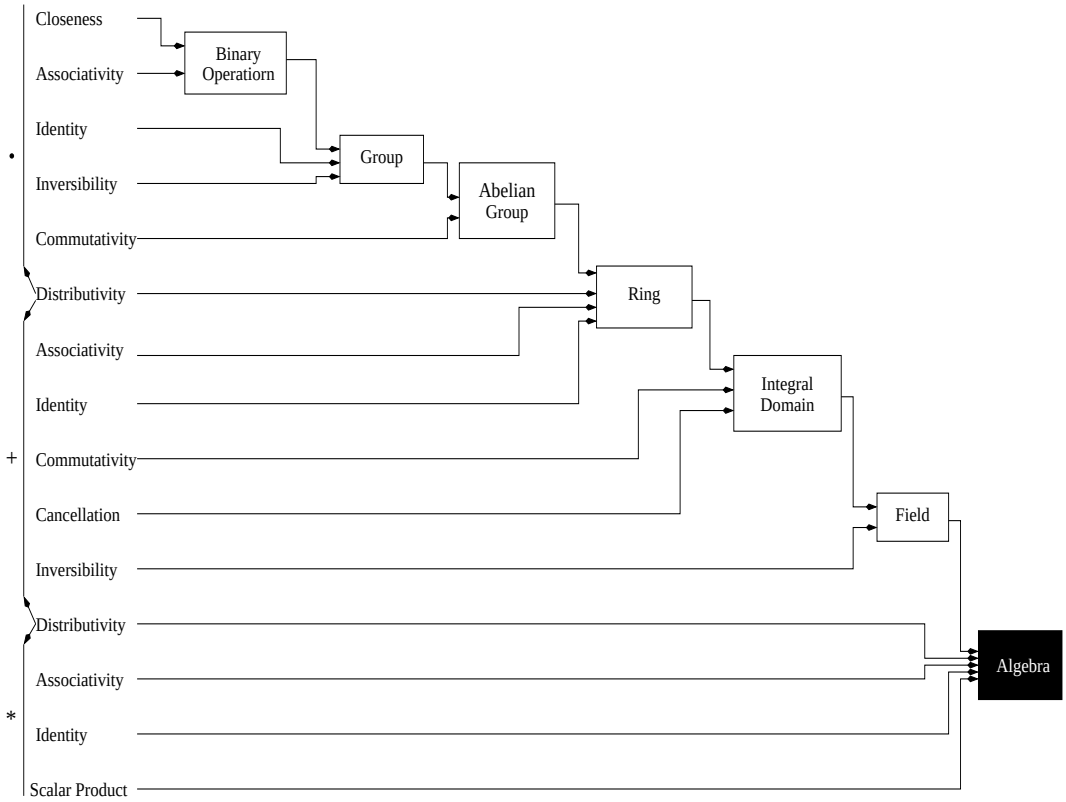

Figure 3.1: Algebraic Structures

Definition 3.26 *Let V and W be vector spaces over a field F. A **linear operator** T is a mapping of the form $T : V \rightarrow W$, such that the following two conditions are satisfied:*

1. *For any vectors $v_0, v_1 \in V$,*

$$T\{v_0 + v_1\} = T\{v_0\} + T\{v_1\} \tag{3.11}$$

2. *For any scalars $a, b \in F$ and $v_0 \in V$,*

$$T\{av_0\} = aT\{v_0\} \tag{3.12}$$

then,

$$T\{av_0 + bv_1\} = aT\{v_0\} + bT\{v_1\} \tag{3.13}$$

The notation $T\{v_0\}$ denotes the action of the operator T over the vector $v_0 \in V$. It results on a new vector that also belongs to the space V.

Definition 3.27 *If V and W are vector spaces over a field F, every linear transformation T of V in W, is called **isomorphism** of V on W. If there exists an isomorphism of V on W, then is said that the sapce V is isomorphic to the space W. All vector space of finite dimension, over a field F, is isomorphic to the space F^n.*

3.2 Finite Dimensional Signal Algebra

Signals are represented mathematically as numeric functions of one or more independent variables, which carry information about the behavior or nature of some physical system [27]. This work is concentrated in the treatment of physical signals that admit a mathematical representation. These signals are finite and discrete, and are considered here as vectors belonging to a complex finite-dimensional Hilbert space. This space, denoted by

$\ell^2(\mathbb{Z}_N)$, is composed of all complex signals of length N, of the followig form:

$$\begin{array}{rccc} x: & \mathbb{Z}_N & \rightarrow & \mathbb{C} \\ & n & \mapsto & x[n] \end{array} \tag{3.14}$$

Where $\mathbb{Z}_N = \{0, 1, 2, \ldots, N-1\}$ and $x = [x[0], x[1], \ldots, x[N-1]]$.

This representation of signals is used to introduce a *finite dimensional signal algebra*, which is defined here as any linear algebra whose elements can be represented as vectors in a finite dimensional Hilbert space, a linear vector space over the scalar body of field of complex numbers $\mathbb{C}$. Thus, a signal algebra is defined in this work as a vector space, such as, $\ell^2(\mathbb{Z}_N)$ with an associated binary operation. This binary operation takes two arbitrary signals, say $x, h \in \ell^2(\mathbb{Z}_N)$, and relates them to a new signal, say $y \in \ell^2(\mathbb{Z}_N)$. In this work, the cyclic convolution operation is used as the binary operation that turns the space $\ell^2(\mathbb{Z}_N)$ into an N-dimensional signal algebra. Thus, all the definitions stated in section 3.1 can be applied on this signal algebra in order to perform operations with signals, such as, addition and scalar multiplication.

In the space $\ell^2(\mathbb{Z}_N)$, the set of any N linearly independent vectors is called *basis*. It is of interest a special finite set of linearly independent vectors denoted by $\left\{\delta_{\{k\}} : k \in \mathbb{Z}_N\right\}$, where:

$$\delta_{\{k\}}[n] = \begin{cases} 1, & k = n \\ 0, & otherwise \end{cases} \tag{3.15}$$

This set forms a basis for the space $\ell^2(\mathbb{Z}_N)$ called the *standard basis set*. This basis is denoted by $\Delta_N = \left\{\delta_{\{0\}}, \delta_{\{1\}}, ..., \delta_{\{N-1\}}\right\}$.

Then, it is established that any signal $x \in \ell^2(\mathbb{Z}_N)$ can be expressed in terms of a linear combination of the standard basis set Δ_N as follows:

$$\begin{array}{rcl} x[n] & = & \displaystyle\sum_{k=0}^{N-1} c[k]\,\delta_{\{k\}} \quad , \quad n \in Z_N \\ & = & c[0]\,\delta_{\{0\}} + c[1]\,\delta_{\{1\}} + ... + c[N-1]\,\delta_{\{N-1\}} \end{array} \tag{3.16}$$

The coefficients $c[k]$'s are called the coefficients of the representation, they are obtained using the fact that the basis set Δ_N is an independent set.

The space $\ell^2(\mathbb{Z}_N)$ has a defined inner product, denoted by $\langle x, y\rangle$, where x and y are two arbitrary signals. Thus, the inner product for x and y is given by:

$$\begin{aligned} \langle\ ,\ \rangle: \quad \ell^2(\mathbb{Z}_N) \times \ell^2(\mathbb{Z}_N) \quad &\to \mathbb{C} \\ (x, y) \quad &\longmapsto \langle\ ,\ \rangle\{(x, y)\} = \langle x, y\rangle = \sum_{k\in\mathbb{Z}_N} x[k]\, y^*[k] \end{aligned} \tag{3.17}$$

Where the symbol $*$ denotes complex conjugation.

Notice that $\langle x, y\rangle \neq \langle y, x\rangle$. It is said that a signal, say x, is orthogonal to another signal, say y, if the following condition is satisfied: $\langle x, y\rangle = 0$. If $x = y$, we have the total energy of the signal x as follows:

$$\begin{aligned} \langle x, x\rangle &= \sum_{k=0}^{N-1} x[k]\, x^*[k] \\ &= \sum_{k=0}^{N-1} |x[k]|^2 \end{aligned} \tag{3.18}$$

Thus, the norm of a finite, discrete signal x, is given by the square root of its total energy as follows:

$$\begin{aligned} \|x\| &= \langle x, x\rangle^{1/2} \\ &= \sqrt{\sum_{k=0}^{N-1} |x[k]|^2} \end{aligned} \tag{3.19}$$

The inner product operation defined on $\ell^2(Z_N)$ is used to obtain the coefficients of the representation. Taking the inner product on both sides of the expression $x = \sum_{k\in Z_N} c[k]\, \delta_{\{k\}}$ we have:

$$\left\langle x, \delta_{\{l\}}\right\rangle = \sum_{k\in Z_N} c[k] \left\langle \delta_{\{k\}}, \delta_{\{l\}}\right\rangle \tag{3.20}$$

The standard basis Δ_N is an orthogonal set, then we have:

$$\left\langle \delta_{\{k\}}, \delta_{\{l\}} \right\rangle = \begin{cases} 0, & k \neq l \\ N, & k = l \end{cases} \tag{3.21}$$

Therefore, using the orthogonality condition of the elements of the standard basis set, it is obtained an expression that defines the projection of a signal with respect to the elements of Δ_N as follows:

$$\left\langle x, \delta_{\{l\}} \right\rangle = c\left[l\right] \left\langle \delta_{\{l\}}, \delta_{\{l\}} \right\rangle \tag{3.22}$$

And the coefficients of the representation of a signal $x \in \ell^2\left(\mathbb{Z}_N\right)$, with respect to the basis Δ_N are given by:

$$c\left[l\right] = \frac{\left\langle x, \delta_{\{l\}} \right\rangle}{\left\langle \delta_{\{l\}}, \delta_{\{l\}} \right\rangle} \tag{3.23}$$

3.3 Signal Algebra Operators on $\ell^2\left(Z_N\right)$

From mathematics, it is known that an operator transform functions into other functions. In other words, an operator maps a function to another. In the signal algebra defined before, we deals with operators who act on the signals to produce other signals. These operators are called *signal algebra operators*. A signal operator on $\ell^2\left(Z_N\right)$, can be viewed in this work as a *system* that accepts as its input a finite-discrete signal, say $x \in \ell^2\left(\mathbb{Z}_N\right)$, and acts on it in order to produce and output signal, say y, that also belongs to the space $\ell^2\left(\mathbb{Z}_N\right)$. Consider figure 3.2 to understand this definition of a signal algebra operator. All the signal operators introduced in this work are defined as actions on signals that belong to the Hilbert space $\ell^2\left(Z_N\right)$, which is isomorphic to the space $\mathbb{C}^N$.

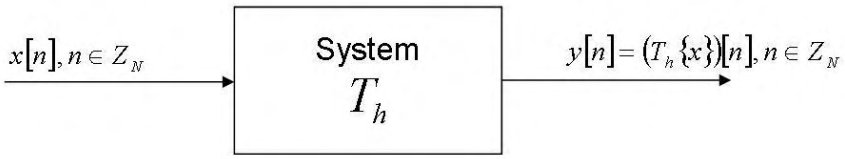

Figure 3.2: Unary Signal Algebra Operator

Mathematically, inputs to an operator receive the name of operands. According to the number of operands that the operators can have, they can be classified as unary, binary, etc. For example, figure 3.2 display the representation of a unary signal operator. In this work, it is given special attention to unary and binary signal algebra operators. Below are presented the general mathematical formulations of unary and binary signal algebra operators on the space $\ell^2(Z_N)$.

3.3.1 Unary Signal Operators

A unary signal operator T_h in $\ell^2(\mathbb{Z}_N)$ is a mapping of the following form:

$$\begin{array}{rccc} T_h: & \ell^2(\mathbb{Z}_N) & \rightarrow & \ell^2(\mathbb{Z}_N) \\ & x & \mapsto & T_h\{x\} = y \end{array} \tag{3.24}$$

In this case, the operand is the signal x who receives the action of the operator T_h.

3.3.2 Binary Signal Operators

A binary signal operator T_h in $\ell^2(Z_N)$ is a mapping of the following form:

$$\begin{array}{rccc} T_h: & \ell^2(\mathbb{Z}_N) \times \ell^2(\mathbb{Z}_N) & \rightarrow & \ell^2(\mathbb{Z}_N) \\ & (x,h) & \mapsto & T_h\{(x,h)\} = y \end{array} \tag{3.25}$$

Thus, it is noticed that a binary operator on $\ell^2(\mathbb{Z}_N)$ takes any two arbitrary signals, say $x, h \in \ell^2(\mathbb{Z}_N)$, and it relates them to a new signal, say $y \in \ell^2(\mathbb{Z}_N)$.

The algebra of the cyclic convolution operation, also called the algebra of the shift operators and the algebra of the Hadamard product operation are two specific algebras of special interest in this work. Therefore, it is necessary to define two important binary operators called cylic convolution and the Hadamard product.

Definition 3.28 *The **cyclic convolution** operator over the space $\ell^2(\mathbb{Z}_N)$ is denoted by the symbol $\circledast_N$, and is a mapping of the form:*

$$\begin{array}{rccl} \circledast_N : & \ell^2(\mathbb{Z}_N) \times \ell^2(\mathbb{Z}_N) & \rightarrow & \ell^2(\mathbb{Z}_N) \\ & (x, h) & \mapsto & \circledast_N \{(x, h)\} = y \triangleq x \circledast_N h \end{array} \tag{3.26}$$

Where $y \in \ell^2(\mathbb{Z}_N)$ results from the cyclic convolution between the signals $x[n]$ and $h[n], n \in Z_N$ as follows:

$$y[n] = \sum_{k=0}^{N-1} x[k] \cdot h[\langle n-k \rangle_N] \quad , \quad n \in Z_N \tag{3.27}$$

Where $\langle n-k \rangle_N$ denotes the remainder of the quotient $\left(\frac{n+k}{N}\right)$.

Definition 3.29 *The **Hadamard product** operator over the space $\ell^2(\mathbb{Z}_N)$ is denoted by the symbol $\odot_N$, and is a mapping of the form:*

$$\begin{array}{rccl} \odot_N : & \ell^2(\mathbb{Z}_N) \times \ell^2(\mathbb{Z}_N) & \rightarrow & \ell^2(\mathbb{Z}_N) \\ & (x, h) & \mapsto & \odot_N \{(x, h)\} = z \triangleq x \odot_N h \end{array} \tag{3.28}$$

where $z \in \ell^2(\mathbb{Z}_N)$ results from the Hadamard product between the signals $x[n]$ and $h[n], n \in Z_N$ as follows:

$$z[n] = x[n] \cdot h[n] \quad , \quad n \in Z_N \tag{3.29}$$

3.3.3 Matrix Representation of Signal Operators

For any operator, which is linear and acting in a N-dimensional vector space, say $\ell^2(\mathbb{Z}_N)$, its matrix representation is given with respect to the standard basis set Δ_N.

Let T_N be a $N \times N$ matrix representing any signal operator, and acting over a signal $x \in \ell^2(\mathbb{Z}_N)$ as follows:

$$\begin{aligned} T_N : \ell^2(\mathbb{Z}_N) &\rightarrow \ell^2(\mathbb{Z}_N) \\ x &\longmapsto y = T_N \end{aligned} \tag{3.30}$$

If we have $x = \sum_{k=0}^{N-1} c[k]\,\delta_{\{k\}}$, for $\delta_{\{k\}} \in \Delta_N$; $k \in Z_N$, then:

$$y = T_N\{x\} = T_N\left\{\sum_{k\in Z_N} c[k]\,\delta_{\{k\}}\right\} = \sum_{k\in Z_N} c[k]\,T_N\left\{\delta_{\{k\}}\right\} \tag{3.31}$$

3.3.3.1 Right Shift Operator $S_N^{(-)}$

Let $S^{(-)}$ be the right shift operator. This operator acting over the ordered standard basis produces the following result:

$$\begin{aligned} S_N^{(-)} : \ell^2(\mathbb{Z}_N) &\rightarrow \ell^2(\mathbb{Z}_N) \\ \delta_{\{k\}} &\longmapsto S_N^{(-)}\left\{\delta_{\{k\}}\right\} = \delta_{\left\{\langle k+1\rangle_N\right\}} \end{aligned} \tag{3.32}$$

Example 3.4 *Right Shift Operator* $S_4^{(-)}$

$$\begin{aligned} S_4^{(-)} : \ell^2(\mathbb{Z}_4) &\rightarrow \ell^2(\mathbb{Z}_4) \\ \delta_{\{k\}} &\longmapsto S_4^{(-)}\left\{\delta_{\{k\}}\right\} = \delta_{\left\{\langle k+1\rangle_4\right\}} \end{aligned} \tag{3.33}$$

The standar basis set for $N = 4$ *is given by:*

$$\Delta_4 = \left\{\delta_{\{0\}}, \delta_{\{1\}}, \delta_{\{2\}}, \delta_{\{3\}}\right\} \tag{3.34}$$

Then, the right shift operator $S_4^{(-)}$ *acting over* Δ_4 *is given as follows:*

$$S_4^{(-)}\{\Delta_4\} = \left\{\delta_{\left\{\langle 0+1\rangle_4\right\}}, \delta_{\left\{\langle 1+1\rangle_4\right\}}, \delta_{\left\{\langle 2+1\rangle_4\right\}}, \delta_{\left\{\langle 3+1\rangle_4\right\}}\right\} = \left\{\delta_{\{1\}}, \delta_{\{2\}}, \delta_{\{3\}}, \delta_{\{0\}}\right\} \tag{3.35}$$

Then,

$$S_4^{(-)} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.36}$$

Example 3.5 $S_4^{(-)}$ *acting over a signal* x

$$\begin{array}{rl} S_N^{(-)} : \ell^2(\mathbb{Z}_N) & \to \ell^2(\mathbb{Z}_N) \\ x & \longmapsto S_N^{(-)}\{x\} = g \end{array} \text{ , where } g[n] = x\left[\langle n-1\rangle_N\right], n \in \mathbb{Z}_N \tag{3.37}$$

Let, $x = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix}$ *then* $S_4^{(-)}$ *acting over the signal* x *is obtained as follows:*

$$S_4^{(-)}\{x\} = g \ , \ g = \begin{bmatrix} g[0] = x\left[\langle 0-1\rangle_4\right] \\ g[1] = x\left[\langle 1-1\rangle_4\right] \\ g[2] = x\left[\langle 2-1\rangle_4\right] \\ g[3] = x\left[\langle 3-1\rangle_4\right] \end{bmatrix} = \begin{bmatrix} g[3] \\ g[0] \\ g[1] \\ g[2] \end{bmatrix} \tag{3.38}$$

In matrix-vector product operation it is obtained:

$$S_4^{(-)} \cdot x = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} = \begin{bmatrix} x[3] \\ x[0] \\ x[1] \\ x[2] \end{bmatrix} \tag{3.39}$$

3.3.3.2 Left Shift Operator $S_N^{(+)}$

Let $S^{(+)}$ be the left shift operator. This operator acting over the ordered standard basis produces the following result:

$$\begin{aligned} S_N^{(+)} : \ell^2(\mathbb{Z}_N) &\to \ell^2(\mathbb{Z}_N) \\ \delta_{\{k\}} &\longmapsto \delta_{\left\{\langle k-1\rangle_N\right\}} = S_N^{(+)}\left\{\delta_{\{k\}}\right\} \end{aligned} \tag{3.40}$$

Example 3.6 *Left Shift Operator* $S_4^{(+)}$

$$\begin{aligned} S_4^{(+)} : \ell^2(\mathbb{Z}_4) &\to \ell^2(\mathbb{Z}_4) \\ \delta_{\{k\}} &\longmapsto S_4^{(-)}\left\{\delta_{\{k\}}\right\} = \delta_{\left\{\langle k-1\rangle_4\right\}} \end{aligned} \tag{3.41}$$

The left shift operator $S_4^{(+)}$ *acting over* Δ_4 *is given by:*

$$S_4^{(+)}\{\Delta_4\} = \left\{\delta_{\left\{\langle 0-1\rangle_4\right\}}, \delta_{\left\{\langle 1-1\rangle_4\right\}}, \delta_{\left\{\langle 2-1\rangle_4\right\}}, \delta_{\left\{\langle 3-1\rangle_4\right\}}\right\} = \left\{\delta_{\{3\}}, \delta_{\{0\}}, \delta_{\{1\}}, \delta_{\{2\}}\right\} \tag{3.42}$$

Then,

$$S_4^{(+)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.43}$$

Example 3.7 *Left shift operator $S_4^{(+)}$ acting over a signal x*

$$\begin{array}{rl} S_N^{(+)} : \ell^2(\mathbb{Z}_N) & \to \ell^2(\mathbb{Z}_N) \\ x & \longmapsto S_N^{(+)}\{x\} = g \end{array} \quad , \text{ where } g[n] = x[\langle n+1 \rangle_N], n \in \mathbb{Z}_N \tag{3.44}$$

$$S_4^{(+)}\{x\} = g \ , \ g = \begin{bmatrix} g[0] = x[\langle 0+1 \rangle_4] \\ g[1] = x[\langle 1+1 \rangle_4] \\ g[2] = x[\langle 2+1 \rangle_4] \\ g[3] = x[\langle 3+1 \rangle_4] \end{bmatrix} = \begin{bmatrix} g[1] \\ g[2] \\ g[3] \\ g[0] \end{bmatrix} \tag{3.45}$$

In matrix-vector product operation it is obtained:

$$S_4^{(+)} \cdot x = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} = \begin{bmatrix} x[1] \\ x[2] \\ x[3] \\ x[0] \end{bmatrix} \tag{3.46}$$

CHAPTER 4

Modeling of Point Spread Functions as Discrete Ambiguity Functions

Modeling and simulation, are tasks for developing a level of understanding of the behavior of a real system, its structure, elements, and processes; to a greater extent than is possible by other educational means. For this reason, this work is formulated in the context of mathematical modeling and system simulation. This chapter presents the theoretical basis of modeling and simulation, as well as, the methodology employed in this research to perform PSFs simulations. In addition, concepts about radar imaging systems, real aperture radar, and synthetic aperture radar are introduced. As a contribution, a model for SAR raw data generation and mathematics of the radar ambiguity function in terms of signal algebra operators are proposed. Finally, the simulations algorithms are presented.

4.1 Introduction to Modeling and Simulation

In this section, modeling and simulation are presented as two separate activities. First, a general perspective of mathematical modeling and its inherent aspects are exposed. Later, simulation is presented as a set of systematically organized steps that describe the execution of a mathematical model. Finally, modeling is presented as an implicit part within the simulation process.

4.1.1 Modeling

In general, modeling can be defined as a way of representing a physical system or entity. In this work, this representation is carried out through a mathematical model, which consists of a set of mathematical structures and equations based on a group of established assumptions, designed to correspond to a physical system. A system, in general is defined here as a set of elements and its interrelationships.

As it is stated by Karplus, the construction of a valid mathematical model requieres the knowledge of the type of elements which are present in the system and how these elements are interconnected [28]. These interconnections represent the way in which the matter or energy flows within the system. Also, a mathematical model is valid only if the system to be modeled satisfies the following three conditions:

- *Separability:* The system to be modeled can be studied as a separate entity, i.e., its elements can be enumerated and some interactions with the external world can be omitted.
- *Selectivity:* Assume that of all possible interactions, only a small subset are relevant to a specific study or purpose. In mathematical terms, the selectivity condition of a system model can be expressed as the following relation: $S \subset X \times Y$. Where S denotes the system being modeled. X and Y denote input and output sets respectively.
- *Causality:* This condition establishes that the input and output sets must be related

by a mapping function, so that $S : X \to Y$. This means that the system behaves like an operator which acts on the input set to produce the output set.

The mathematical expressions that describes the system model, is composed of variables which represents some properties of the system. The values assigned to these variables can be of any types; real or complex numbers, logical values or strings, for example. The parameters are numerical values assigned to the coefficients appearing in the equations, which generally are related to the magnitudes of the elements constituting the system.

There are different types of mathematical models, they can be deterministic or stochastic. Deterministic models perform the same way for a given set of initial conditions. While stochastic models present randomness, even when an identical set of initial conditions is given. A mathematical model handling with continuous variables is called a continuous model. In another way, if the variables associated to the mathematical model are discrete, then it is a discrete model.

4.1.2 Simulation

Once the mathematical model has been developed, the parameters have been fixed, and the initial conditions have been established; is of interest to see the system model in operation. In few words, when it is talked about the model in action, it is referred to the meaning of simulation.

Another point of view of simulation, is presented by Fishwick [29]. Here, simulation is viewed as the composition of the following three processes: model design, model execution, and execution analysis. This interpretation is illustrated in figure 4.1. Notice that the modeling process is considered as part of the simulation process. The model is executed through a program running on a computer. This program can be implemented in serial or parallel modes.

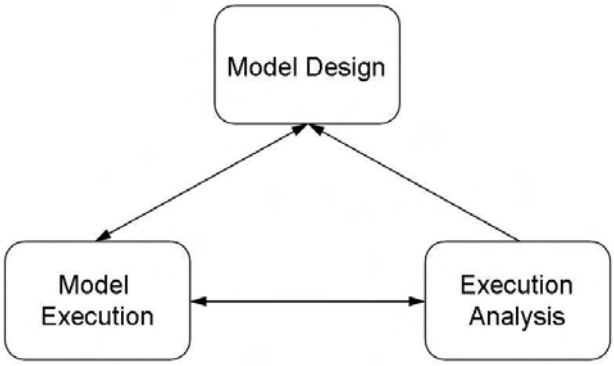

Figure 4.1: The Simulation Process

4.2 Radar Imaging Systems

Radar imaging systems can be described in few words as devices carried on an aircraft or spacecraft platform at uniform speed and altitude, which transmit a series of electromagnetic pulses to illuminate an area on the Earth surface. These systems can operate at different bands in the microwave region of the electromagnetic spectrum as is displayed in table 4.1. The system antenna emits a narrow beam of microwave pulse signals directed perpendicularly to the flight path of the carrier platform. Then, the reflected energy is used to form an image of a narrow strip of the ground.

An important parameter of any imaging system is its *spatial resolution*, which is defined as the minimun distance at which two different objects are detected by the system as separated [16]. The spatial resolution of a radar imaging system is governed by two important spects: the length of the transmitted pulse signal and the antenna beam width. The pulse length determines resolution in the direction in which the signals are transmitted. This is called the *range direction*. In the other hand, the antenna beam width determines resolution in the direction in which the carrier platform flies. This is called *azimuth direction*. Thus, the produced images are composed of rectangles called resolution cells whose sides

are given by a range component, say ΔR; and an azimuth component, say ΔA. Figure 4.2 depicts a graphical representation of a map resolution cell.

Table 4.1: Radar Operational Frequency Bands

Letter designation	Frequency (GHz)	New band designation (GHz)
HF	$0.003 - 0.03$	A
VHF	$0.03 - 0.3$	A<0.25; B>0.25
UHF	$0.3 - 1.0$	B<0.5; C>0.5
L-band	$1.0 - 2.0$	D
S-band	$2.0 - 4.0$	E<3.0; F>3.0
C-band	$4.0 - 8.0$	G<6.0; H>6.0
X-band	$8.0 - 12.5$	I<10.0; J>10.0
Ku-band	$12.5 - 18.0$	J
K-band	$18.0 - 26.5$	J<20.0; K>20.0
Ka-band	$26.5 - 40.0$	K
MMW	Normally>34.0	L<60.0; M>60.0

Radar imaging systems are classified into two main categories: real aperture radar (RAR) and synthetic aperture radar (SAR) systems. The main difference between these systems reside in their spatial resolution capability. Range resolution is the same for both systems, and depends on the pulse width of the transmitted signal as shown in figure 4.3. In this figure θ is the angle of incidence, τ_p is the pulse width or pulse duration, and C is the speed of light. Obtaining high range resolution is possible by transmitting a pulse with a very short duration and high-power peak. However, the required equipment to transmit a very short, high-energy pulse is difficult to build. Therefore, these systems use pulse compression techniques by frequency modulation. For this, a special waveform is transmitted, such as, a linear frequency modulated (LFM) or chirp signal. The received signal is processed through a matched filter in order to make the pulse width to a shorter value.

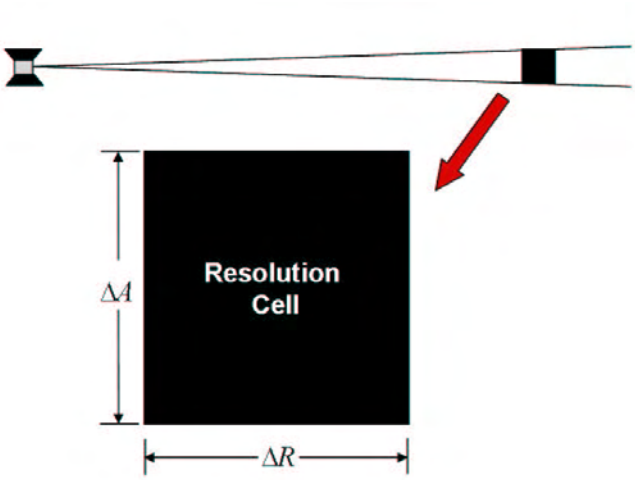

Figure 4.2: Resolution Cell of a Radar Imaging System

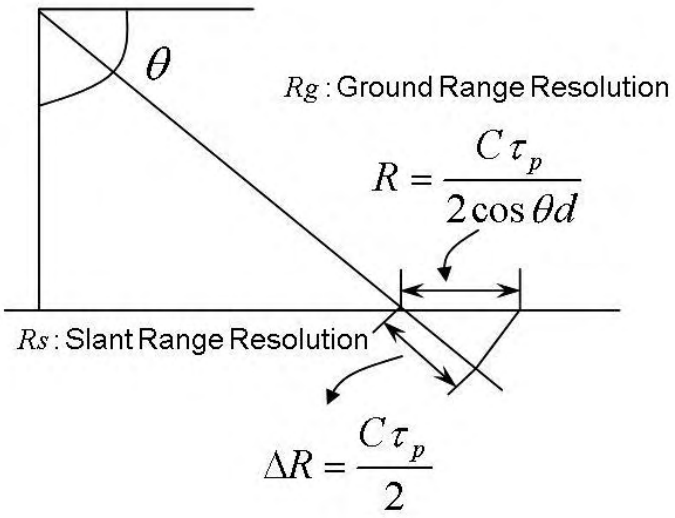

Figure 4.3: Range Resolution

In general, a RAR system, also called SLAR (Side Looking Airborne Radar) employs a fixed antenna length for transmission and reception, and the backscattered signal are received from the same location as the initial transmitted signal. The azimuth or cross-range resolution for a real antenna is determined by the antenna beam width and the distance between the radar and the target (slant range) as shown in figure 4.4. Where β is the antenna beam width, λ is the wavelength, D is the aperture length, and ΔA_1 and ΔA_2 are azimuth resolutions for target 1 and 2 respectively.

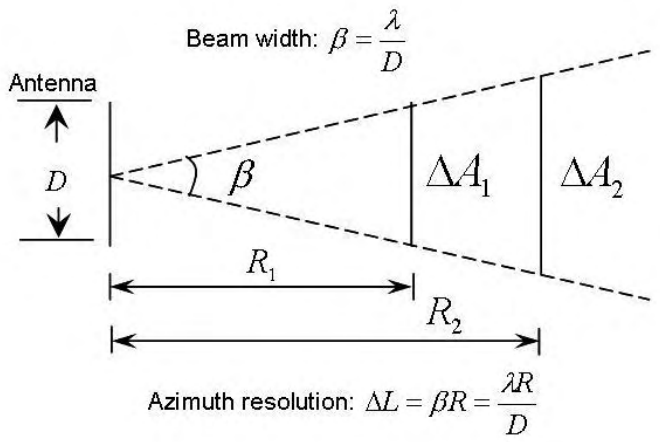

Figure 4.4: Azimuth Resolution

In order to obtain high azimuth resolution, short wavelengths and a big antenna size are needed. However, it is hard to fix a large antenna on an airborne or spaceborne platform. For example, a $1Km$ diameter antenna is required to obtain $25m$ of azimuth resolution with a wavelength of $25cm$ (L band) and $100Km$ distance from the target. Therefore, RAR systems present a technical limitation for increasing azimuth resolution.

4.2.1 Synthetic Aperture Radar (SAR)

SAR technology systems were developed as a solution to the problem of limited resolution present in RAR systems. A synthetic aperture is produced by using the forward motion of the system platform. The radar transmits and receives from different positions. The reflected pulses and phases are recorded and combined, such that, by means of signal processing techniques SAR can synthesise an aperture that is longer than the real.

The effective synthetic aperture length is twice that of a real array, the azimuth resolution for a synthetic array is then given by [30]:

$$\Delta A = \frac{\lambda R}{2L} \tag{4.1}$$

Where L is the synthetic aperture length.

Azimuth resolution can be impoved by taking advantage of the frequency variation or Doppler history within a beam. Let $R(t)$ denote the range to a scatterer at time t, and v_r be the corresponding radial velocity; thus the Doppler shift, f_d, is given by:

$$f_d = -\frac{2R'(t)}{\lambda} = \frac{2v_r}{\lambda} \tag{4.2}$$

Where $R'(t)$ corresponds to the range rate to a scatterer. From figure 4.5, t_1 and t_2 are times when the scatterer enters and leaves the radar beam, respectively, and t_c is the time that corresponds to minimum range. Doppler frequency is maximum at t_1, zero at t_c, and minimum at t_2.

From the geometry of figure 4.5 the maximum Doppler frequency at t_1 and the minimun Doppler frequency at t_2 becomes [30]:

$$f_{d_{\max}} = \frac{2v}{\lambda} \cos\left(90 - \frac{\theta}{2}\right) \sin\beta \tag{4.3}$$

$$f_{d_{\min}} = \frac{2v}{\lambda} \cos\left(90 + \frac{\theta}{2}\right) \sin\beta \tag{4.4}$$

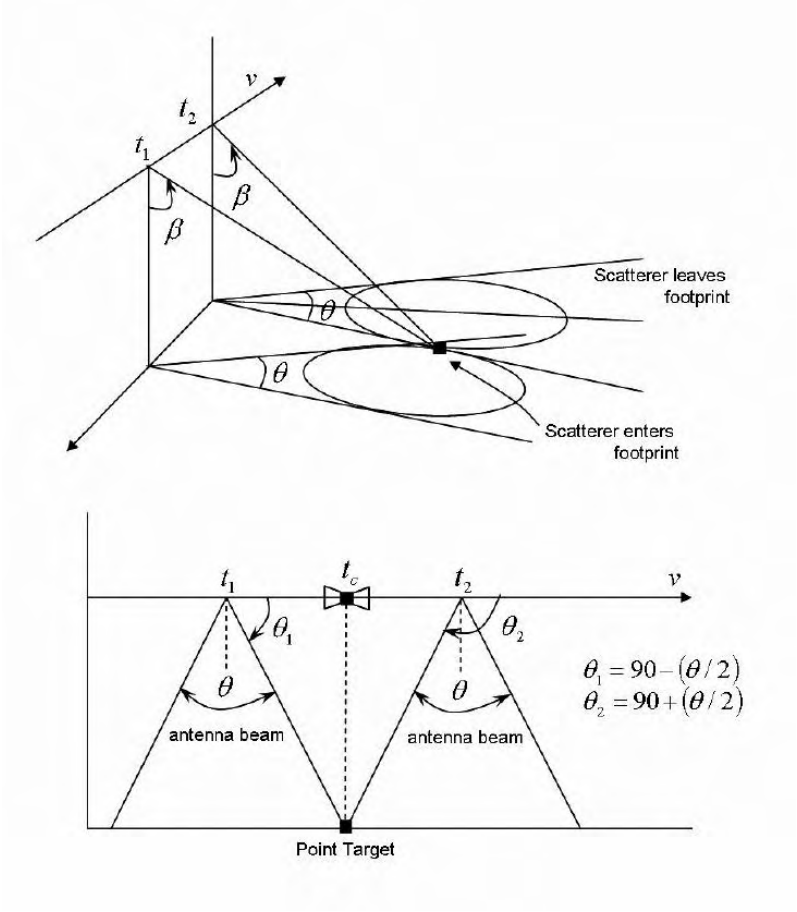

Figure 4.5: Doppler Variation Computation

Thus, the maximum Doppler spread is given by:

$$\Delta f_d = f_{d_{\max}} - f_{d_{\min}} \tag{4.5}$$

After substituting (4.3) and (4.4) in (4.5), the maximum Doppler history equation obtained is:

$$\Delta f_d = \frac{2v}{\lambda}\theta \sin\beta \tag{4.6}$$

It is possible to resolve two adjacent points or scatterers at the same range, based on the difference between its Doppler histories. For this, is assumed that the two scatterers are within the kth range bin. Using equation (4.6) the minimum Doppler spread between the scatterers is given by:

$$\Delta f_{d_{\min}} = \frac{2v}{\lambda}\Delta\theta \sin\beta_k \tag{4.7}$$

Where $\Delta\theta$ is the angular displacement between the scatterers, and β_k is the elevation angle corresponding to the kth range bin. The synthetic aperture length L is equal to vT_{ob}, where T_{ob} is the coherent integration interval. Then, substituting L in equation (4.7) and solving for $\Delta\theta$ obtain the following:

$$\Delta\theta = \frac{\lambda}{2L\sin\beta_k} \tag{4.8}$$

Finally, the SAR azimuth resolution within the kth range bin is given by:

$$\Delta A = \Delta\theta R_k = R_k \frac{\lambda}{2L\sin\beta_k} \tag{4.9}$$

4.2.1.1 Mathematical Model of Radar Imaging Systems

The mathematical model of radar imaging systems considered in this work is the fomulated by R. Blahut on his work on remote surveillance algorithms [5]. Figure 4.6 depicts this model.

Let $\gamma(\tau, v)$ be the reflectivity density function of an input scene to a radar imaging system. The expected output of the system, $I(\tau, v)$, is obtained by performing the two-dimensional convolution of the reflectivity density function and the impulse response

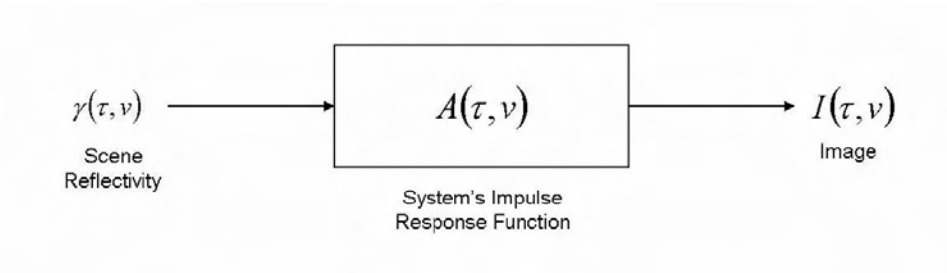

Figure 4.6: Blahut Model of Imaging Radar Systems

function of the system as follows:

$$I(\tau, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \gamma(\tau', v') A(\tau - \tau', v - v') d\tau' dv' \tag{4.10}$$

The system impulse response function or PSF of a radar imaging system is modeled as the radar ambiguity function between the transmitted and received radar signals, denoted by $A(\tau, v)$.

Suppose that the two-dimensional impulse response, $A(\tau, v)$, be equal to a two-dimensional delta Dirac impulse function:

$$A(\tau, v) = \delta(\tau, v) \tag{4.11}$$

Then, the image I would be equal to the reflectivity density function of the observed scene. In practice, the impulse response function introduce degradations and blur that affect the appearance of the output image. For this reason, it must be processed in order to retrieve the scene reflectivity.

4.2.2 SAR Imaging System Model

The SAR imaging system model is composed of three main stages: point spread function, raw data generation and image formation. This work, is concentrated on the point spread function modeling and simulation. However, models for SAR raw data generation

and SAR image formation are formulated as follows:

- *Raw data generation:* In general the SAR raw data generation system accepts as input the scene reflectivity and produce as output the raw data image. This system is represented in figure 4.7.

Figure 4.7: SAR System Model

In this work, the mathematical model of a radar imaging system proposed by Blahut is used to formulate a model for SAR raw data generation. Following the Blahut formulations, the raw data is the output of a radar imaging system which is obtained from the two-dimensional cyclic convolution of the scene reflectivity with the system impulse response function. The raw data generation model is depicted in figure 4.8. The impulse response function or PSF of this model is the ambiguity function between radar transmitted and received signals. In this model it is assumed that the SAR system is discrete, space-invariant and linear.

s_t and s_r correspond to the transmitted and received radar signals respectively, the symbol $\circledast_2$ denotes two-dimensional cyclic convolution, $\gamma[n, \tau, \theta]$ is the reflectivity density function, $g[l_x, l_y]$ is the SAR raw data, and $A[m, k]$ is the discrete ambiguity

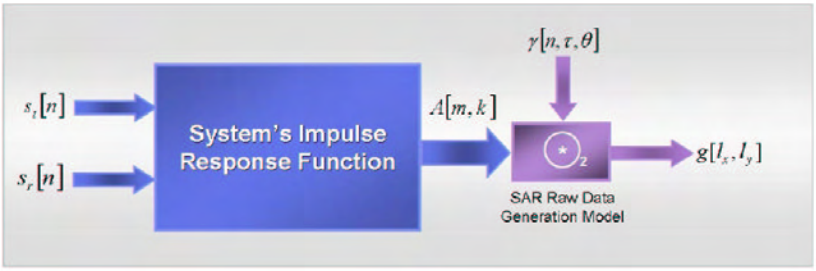

Figure 4.8: SAR Raw Data Generation Model

function which will be described in later sections. Raw data is also called level zero data.

From figure 4.7, notice that raw data is a noisy image that not provides visual information about the original image. This occurs since each point in the raw data image is the result of the convolution between the entire two-dimensional impulse response function with all the information contained in the reflectivity density function as presented in equation (4.10).

- *Image formation:* Image formation consists of processing the level zero data in order to extract the surface reflectivity information corresponding to a given area on the ground. The model used in this work is a model proposed by Franceschetti [16]. Figure 4.9 illustrates this model. In this model, the raw data is available but the system impulse response is unknown. Then, this process must be treated as an inverse problem in order to recover the reflectivity of the scene.

From the cyclic convolution theorem, raw data becomes:

$$G[\xi,\eta] = (\Gamma \odot H)[\xi,\eta] \tag{4.12}$$

Where H is the system transfer function or the Fourier transform of the estimated value of the impulse response function. Γ is the Fourier transform of the reflectivity.

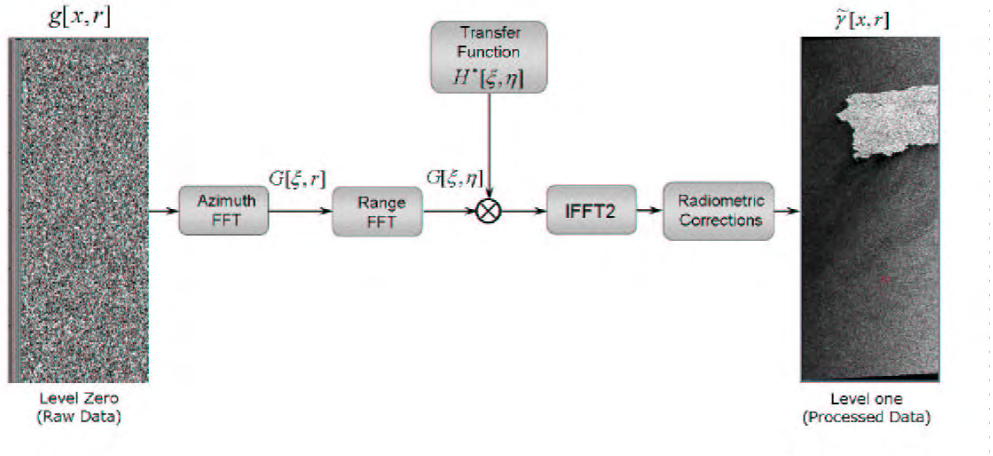

Figure 4.9: SAR Image Formation Model

x and r represents azimuth and range respectively in time domain. ξ and η represents azimuth and range in space domain.

To estimate the reflectivity of the scene from the equation 4.12:

$$\Gamma\left[\xi,\eta\right] = \left(G \odot H^{*}\right)\left[\xi,\eta\right] \tag{4.13}$$

Therefore, the estimated reflectivity density function is obtained using the inverse two-dimensional Fourier transform as follows:

$$\tilde{\gamma}\left[x,r\right] = FFT2^{-1}\left(\left(G \odot H^{*}\right)\left[\xi,\eta\right]\right) \tag{4.14}$$

Where H^{*} is the complex conjugate of of the system transfer function.

4.3 The Radar Ambiguity Function

The ambiguity function defined in this work can be viewed as a generalized signal autocorrelation tool to simultaneously estimate time delay and Doppler frequency offset parameters between a transmitted radar signal and its returned echo or delayed version of

the transmitted signal. This function accepts the transmitted and received signals as input, and generates a two-dimensional surface, one dimension being time and the other frequency. This simultaneous parameters determination allows for a time-frequency representation of this tool. These parameters can, in turn, be used to estimate, through basic algebraic transformations, range and cross-range (azimuth) parameters in a spatial object domain. The combined range and cross-range differentials or parameter increments determine a given unit resolution cell for a particular point in the spatial object domain. To obtain the range and velocity, the maximum peak in the ambiguity surface is found. The time and frequency that correspond to this peak in the ambiguity surface are the time delay and frequency shift of the received signal.

Following the formulations of Blahut, it is stablished in this work that the ambiguity function is the two-dimensional point spread function for the two-dimensional scenes observed by SAR systems.

The **ambiguity function** for a finite-energy signal $s(t)$ is defined as the absolute value of its two-dimensional correlation function denoted by $|A(\tau, \upsilon)|$:

$$|A_s(\tau, \upsilon)| = \left| \int_{-\infty}^{\infty} s(t)\, s^*(t+\tau)\, e^{-j2\pi\upsilon t} dt \right| \tag{4.15}$$

The ambiguity function can be represented in two forms as follows:

- Symmetrical Form:

$$A(\tau, v) = \int_{-\infty}^{\infty} s(t+\tau/2)\, s^*(t-\tau/2)\, e^{-j2\pi v t} dt \tag{4.16}$$

- Asymmetrical Form:

$$A(\tau, v) = \int_{-\infty}^{\infty} s(t)\, s^*(t-\tau)\, e^{-j2\pi v t} dt \tag{4.17}$$

In this work, the asymmetrical form was used to model SAR point spread function. The properties of the ambiguity function are presented as follows:

1. *Maximum property:*

   The maximum value of the ambiguity function always occurs at the origin, i.e, when $(\tau, \upsilon) = (0, 0)$, and is equal to the total energy of the signal, E:

$$|A_s(\tau, \upsilon)| \leq |A_s(0, 0)| = E \tag{4.18}$$

   Where E is given by:

$$E = \int_{-\infty}^{\infty} |s(t)|^2 \, dt \tag{4.19}$$

2. *Volume property:*

   The total volume under the ambiguity function squared is a constant, equal to the square of the signal energy,

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |A_s(\tau, \upsilon)|^2 \, d\tau d\upsilon = |A_s(0, 0)|^2 = E^2 \tag{4.20}$$

3. *Symmetry property:*

   The ambiguity function is symmetric with respect to the origin,

$$|A_s(\tau, \upsilon)| = |A_s(-\tau, -\upsilon)| \tag{4.21}$$

4. *Scaling property:*

   Let $s'(t) = s(at)$, then the ambiguity function of $s'(t)$ becomes,

$$|A_{s'}(\tau, \upsilon)| = \frac{1}{a} A_s(a\tau, \upsilon/a) \tag{4.22}$$

5. *Quadratic phase property:*

If $s(t)$ is multiplied by a quadratic phase low (linear frequency), $s'(t) = s(t)\, e^{j\pi\alpha t^2}$, then the ambiguity function of $s'(t)$ is given by:

$$|A_{s'}(\tau, \upsilon)| = |A_s(\tau, \upsilon + \alpha\tau)| \tag{4.23}$$

Proofs of the different properties of the radar ambiguity function are provided by Blahut in [5].

4.3.1 Ambiguity Function of Basic Waveforms

1. *Single Pulse Ambiguity Function*

   The normalized rectangular pulse $s(t)$ is defined by:

$$s(t) = \frac{1}{\sqrt{\tau'}} Rect\left(\frac{t}{\tau'}\right) \tag{4.24}$$

   Substituting (4.24) in equation (4.15) and performing integration obtain the ambiguity function of a rectangular pulse:

$$|A(\tau, v)| = \left| \left(1 - \frac{|\tau|}{\tau'}\right) \frac{\sin\left(\pi v\left(\tau' - |\tau|\right)\right)}{\pi v\left(\tau' - |\tau|\right)} \right| ; |\tau| \leq \tau' \tag{4.25}$$

   Figure 4.10 displays a representation of the ambiguity function of a single pulse.

2. *Coherent Pulse Train Ambiguity Function*

   A normalized individual pulse $s(t)$, with pulse width denoted by τ' and pulse repetition interval (PRI) denoted by T, is given by the following expression:

$$s(t) = \frac{1}{\sqrt{N}} Rect\left(\frac{t}{\tau'}\right) \tag{4.26}$$

   Then, normalized coherent train of N rectangular pulses is given by:

$$s(t) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} s_1(t - iT) \tag{4.27}$$

Substituting 4.27 in 4.15 and interchanging the summations and integration obtain:

$$A(\tau, v) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \int_{-\infty}^{\infty} s_1(t - iT)\, s_1^*(t - jT - \tau)\, e^{-j2\pi vt} dt \tag{4.28}$$

After some changes of variables and mathematical manipulations the ambiguity function associated with the coherent pulse train becomes:

$$A(\tau, v) = \frac{1}{N} \sum_{q=-(N-1)}^{N-1} |A_1(\tau - qT, v)| \left| \frac{\sin[\pi v N - |q| T]}{\sin(\pi v T)} \right| ; \tau' < T/2 \tag{4.29}$$

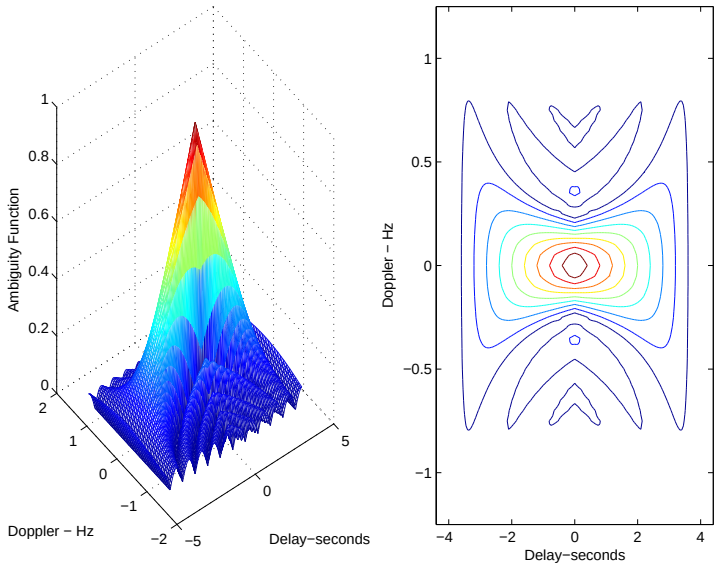

Figure 4.10: The Ambiguity Function of a Single Pulse ($\tau = 4$). 3-D plot (left). Contour plot (right)

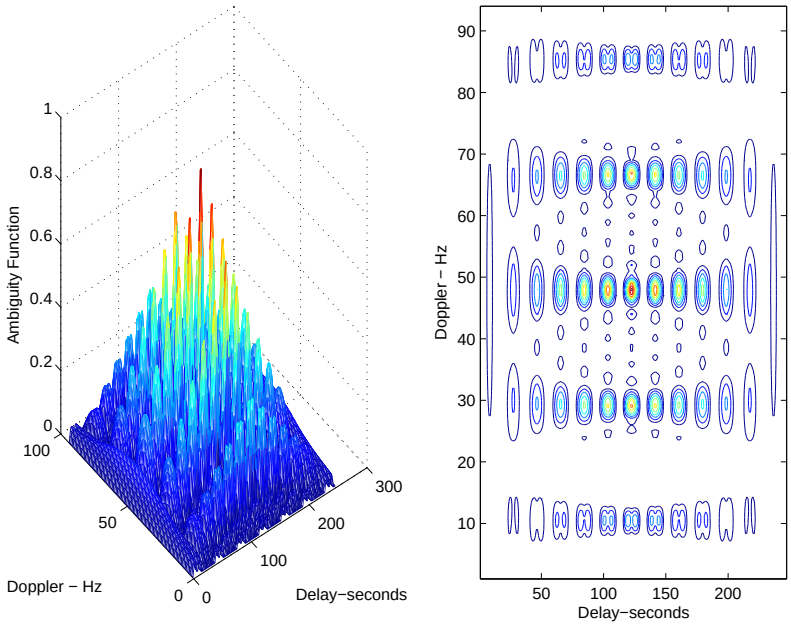

Figure 4.11: The Ambiguity Pulse of 7 Coherent Pulses ($\tau = 0.4$ and $PRI = 1$).3-D plot(left). Contour plot (right)

4.3.2 Discrete Ambiguity Function (DAF)

Let $s_t \in \ell^2(\mathbb{Z}_L)$ and $s_r \in \ell^2(\mathbb{Z}_M)$ be the transmitted and received signals, respectively, of a SAR system after the signals have been demodulated, quatized, and stored in vectors of finite length.

It is defined the discrete cross-ambiguity function of s_t and s_r as follows:

$$\begin{array}{rccl} \mathrm{A}{:}\ell^2(\mathbb{Z}_L) \times \ell^2(\mathbb{Z}_N) & \longrightarrow & \ell^2(\mathbb{Z}_L \times \mathbb{Z}_N) & ; N = L + M - 1 \\ (s_t, s_r) & \longmapsto & A\{(s_t, s_r)\} & \end{array} \tag{4.30}$$

The discrete ambiguity function is a linear correlation between the transmitted and received radar signals. Its mathematical representation is given by:

$$A\{s_t, s_r\}[m,k] = \sum_{n=0}^{L-1} s_t[n]\, s_r^*[m+n]\, W_L^{kn} \;;\; k \in \mathbb{Z}_L, m \in \mathbb{Z}_M \tag{4.31}$$

Where, $W_L = e^{-j\frac{2\pi}{L}}$. Instead of a linear correlation, the discrete ambiguity function can be computed as a cyclic correlation operation. This is possible using the zero-padding operator which turns a linear correlation into a cyclic correlation. The zero-padding operator acts on transmitted and received signals as follows:

$$\begin{aligned} P(L,N) : \ell^2(\mathbb{Z}_L) &\longrightarrow \ell^2(\mathbb{Z}_N) \\ s_t &\longmapsto P\{s_t\} = s'_t \end{aligned} \tag{4.32}$$

$$\begin{aligned} P(L,N) : \ell^2(\mathbb{Z}_L) &\longrightarrow \ell^2(\mathbb{Z}_N) \\ s_r &\longmapsto P\{s_r\} = s'_r \end{aligned} \tag{4.33}$$

Where s'_t and s'_r are the padded transmitted and received signals. Then, the cyclic ambiguity function is obtained as showed in Figure 4.12.

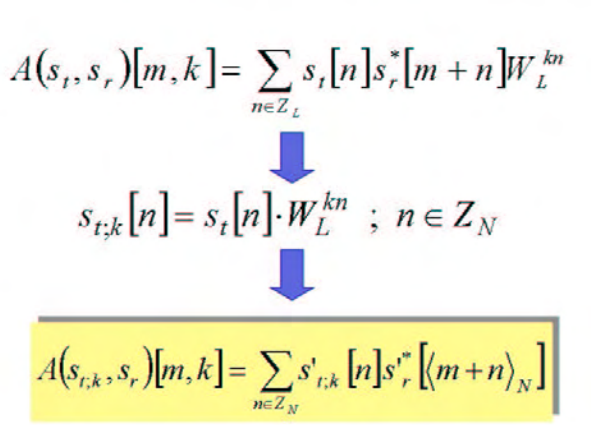

Figure 4.12: Discrete Ambiguity Function as a Cyclic Correlation

The representation of Figure 4.12 corresponds to a method for computing the ambiguity function, called the *filter method*. This will be discussed in the next section.

Let ς be the index-reversal or reflection operator. The cyclic correlation can be turned into a cyclic convolution through the action of the index-reversal operator on one of

the signals as follows:

$$\begin{aligned} \varsigma : \ell^2(\mathbb{Z}_N) &\longrightarrow \ell^2(\mathbb{Z}_N) \\ s'_{t;k} &\longmapsto \varsigma\left\{ s'_{t;k} \right\} = s'_{t;k}[-n] \end{aligned} \tag{4.34}$$

Then, the discrete ambiguity function becomes:

$$A\{s_{t;k}, s_r\}[m,k] = \sum_{p=0}^{N-1} s'_{t;k}[p]\, s'^{*}_{r}[\langle m-p \rangle_N] \;\; ; \;\; k \in \mathbb{Z}_N, m \in \mathbb{Z}_N \tag{4.35}$$

Figure 4.13 shows the relation of linear and cyclic conditions through the implementation of the zero-padding and the index-reversal operators.

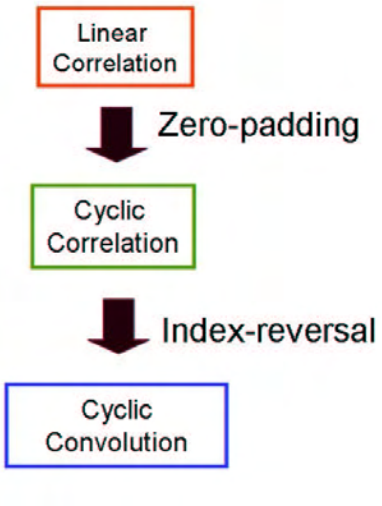

Figure 4.13: Index-Reversal and Zero-Padding Operators

4.4 Algorithms for DAF Computation

The algorithms used in this thesis for SAR point spread functions simulations were introduced in chapter two, from the works presented by D. Rodriguez [31] and R. Tolimieri [7]. These algorithms are based on two methods described below:

1. *DFT Method*

This method allows the computation of the cross-ambiguity function directly by means of the discrete Fourier transform (DFT), as follows: From the general formulation of discrete ambiguity function in equation (4.30), let,

$$v_m[n] = s_t[n] \cdot s_r^*[\langle n+m \rangle_N] \quad ; \; n \in \mathbb{Z}_N \tag{4.36}$$

Then,

$$A\{s_t, s_r\}[m,k] = \sum_{n \in \mathbb{Z}_N} v_m[n] W_L^{kn} = V_m[k] \tag{4.37}$$

Figure 4.14 shows the DFT method algorithm as a composition of signal algebra operators.

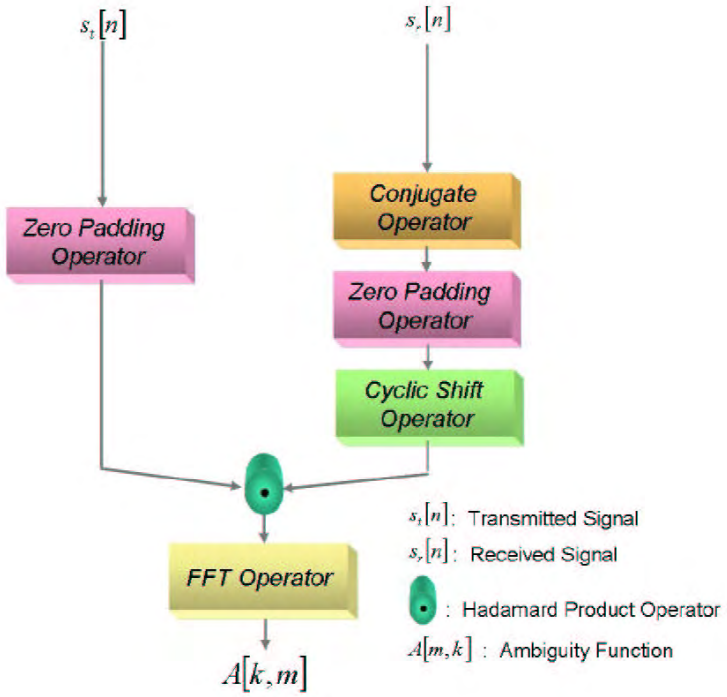

Figure 4.14: DFT Method Algorithm for Ambiguity Function Simulations

2. *Filter Method* The filter method results when the cross-ambiguity function is viewed as a cyclic correlation. This cyclic correlation can turned into a cyclic convolution operation through the index reversal of a weighted version of the transmitted signal s_t, as presented in equation (4.38) below:

$$h_k[n] = s_t[n] W_N^{kn} \quad ; \quad n \in \mathbb{Z}_N. \tag{4.38}$$

The cyclic convolution can, in turn, be computed in an efficient manner through indirect methods using fast Fourier transform (FFT) algorithms. Thus the object-domain correlation version of the cross-ambiguity function is obtained as follows:

$$A\{s_t, s_r\}[m,k] = \sum_{n \in \mathbb{Z}_N} h_k[n] s_r^*[\langle n+m \rangle_N] \tag{4.39}$$

Where the signal h_k can be thought of, after index reversal, as the impulse response signal of a finite impulse response (FIR) filter. Under this condition the coefficients can be pre-computed to reduce the processing effort. Finally, this view allows the realization of a hardware implementation for real time processing. Due to the commutative property of the convolution operation, the return signal $s_r[n]$ could be used as filter coefficients; but it would require signal data buffering impeding the real time realization.

Figure 4.15 shows the filter method algorithm as a composition of signal algebra operators.

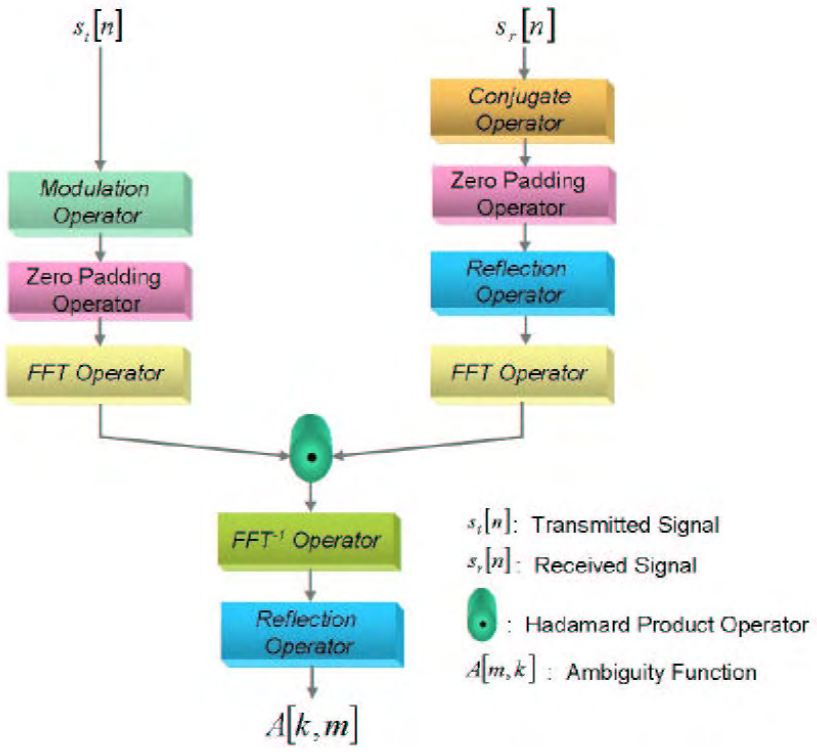

Figure 4.15: Filter Method Algorithm for Ambiguity Function Simulations

4.4.1 Analysis of Complexity

The computational complexity of the DFT method is analized in this part. For this, the DFT method will be presented as a sequence of computational steps as follows:

1. Step 1: From equation (4.36) it is clear that for each m, compute the N products:

$$v_m[n] = s_t[n] \cdot s_r^*[\langle n+m \rangle_N] \quad ; \; n, m \in \mathbb{Z}_N \tag{4.40}$$

   It is assumed here that $N = 2^r$.

2. Step2: For each $0 \leq m \leq N$, compute the N-point one-dimensional FFT as follows:

$$A_{s_t,s_r}[m,k] = \sum_{n \in \mathbb{Z}_N} v_m[n] W_L^{kn} = V_m[k] \tag{4.41}$$

$$V_m[k] = \sum_{n=0}^{N-1} v_m[n] e^{-j\frac{2\pi kn}{N}} \tag{4.42}$$

Step 1 clearly requires $N \times N$ multiplications, but $N = 2^r$, then the number of requiered multiplications is $N^2 = (2^r)^2 = 2^{2r}$. Hence, Step 2 requieres $N^2 logN$ multiplications and additions, because the overall computational complexity of the FFT algorithm is $NlogN$.

Summarizing, the DFT method requieres $N^2 logN$ calculations. Using big-O notation the order of the algorithm is: $O(N^2 logN)$. Thia implies a $O(N^2 logN)$ time which is efficient since it is a polinomial order.

CHAPTER 5

Simulation Results

This chapter presents the results obtained from the modeling and simulation of point spread functions as discrete ambiguity functions of synthetic aperture radar (SAR) imaging systems.

5.1 Results on Modeling

As first result we have the discrete model created for the synthetic aperture radar (SAR) imaging system based on the discrete ambiguity function as its impulse response function. Figure 5.1 depicts this model, which consists of two main stages: raw data generation and image formation.

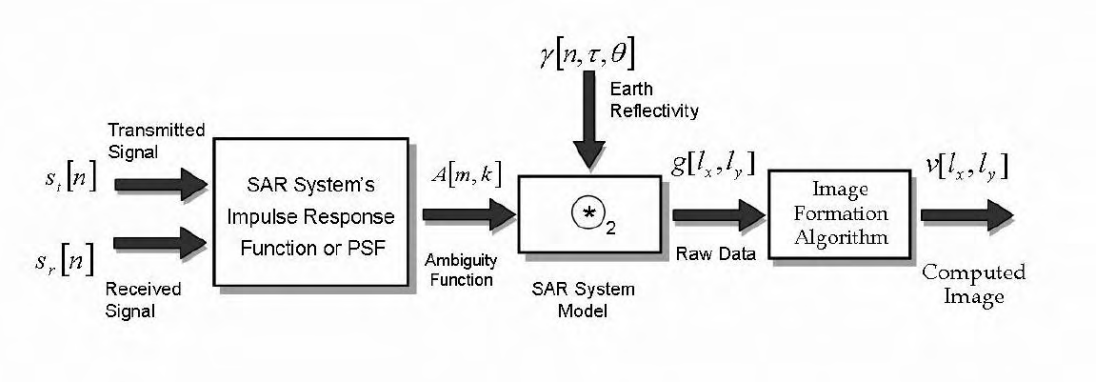

Figure 5.1: SAR imaging system

5.2 MATLAB Simulations Results

In this section, results on MATLAB simulations are presented. The computational methods described in chapter 4 were programmed in MATLAB in order to perform point spread functions simulations through the computation of the discrete ambiguity function. The algorithms were programmed using the *functional programming approach*. This is a style of programming that envisages computation as the process of applying functions to arguments [32]. With this approach, operators can be viewed as functions that receive an input and deliver an output. This output is given by the action of the function over the input arguments. In general, functions are the basic building blocks constructing the structure of this kind of programs. The following example presents the index-reversal operator programmed in MATLAB as a function.

Example 5.1 *Index-reversal operator*

```
function reversedsignal = indexrevop(x,N)
  if length(x) ~= N
    error('Second input parameter does not correspond
           to the true signal length')
      else
        for k=0:N-1;
          reversedsignal(k+1)=x(N-k);
        end
  end
```

In this example, the *indexrevop* is the function that performs the index-reversal operation. The input parameters are the signal, x, and its respective length, N. The output signal is returned in the "*reversedsignal*" array.

One of the advantages of functional programming is that it allows the creation of well-structured programs easy to debug. In addition, contributes with modularity since it provides a set of modules that can be re-used to reduce future programming costs [32].

MATLAB simulations were carried out on a PC workstation with one Pentium III processor running at 800 MHz, with 392,740 KB of RAM, a 9.76 GB hard disk, and

Microsoft Windows 2000 operating system. Table 5.1 presents the results obtained from PSF simulations on MATLAB in terms of the time (in seconds) spent by the system CPU on MATLAB processes during the execution of the programs developed for each computational method.

Table 5.1: Time Measurements of Matlab Simulations

CPU Time (sec)			
Image Size	Cyclic DFT Method	Filter Method	Linear DFT Method
8×23	0.0100	0.1600	0.2000
16×47	0.0200	0.2000	0.2100
32×95	0.0400	0.3300	0.22100
64×191	0.0800	0.5110	0.2200
128×383	0.1510	1.5630	0.3500
256×767	0.3400	9.4330	0.6410
512×1535	1.2920	77.5510	1.5120
1024×3071	5.1970	575.9690	5.7290
2048×6143	26.0560	Out of memory	24.8650
4096×12287	Out of memory	Out of memory	Out of memory

From table 5.1, notice that the DFT method was faster than the filter method for both cases, linear and cyclic. For image sizes greater than 2048×6143 the MATLAB environment is out of memory. Therefore, scalability in this simulations is limited due to the large memory consumption.

5.2.1 Graphical Results of MATLAB Simulations

The results of PSFs simulations obtained from the implementation of the computational methods can be displayed thanks to the visualization tools of MATLAB. As first result, consider figure 5.2 which displays the ambiguity function corresponding to a transmitted pulse signal of 256 sec of duration and a received signal with time delay of 320 sec and frequency-shift equal to 130 Hz. This figure shows a three-dimensional representation of the ambiguity surface. Figures 5.3 and 5.4 shows a two-dimensional representation and a contour diagram respectively from this result. These results were obtained from simulations with the cyclic DFT method.

The following results were obtained from the ambiguity function computation of a transmitted chirp signal of 512 sec of duration and a received signal with time delay of 875 sec and a frequency-shift equal to 230 Hz. Figure 5.5 shows the three-dimensional representation of the resulting ambiguity surface. Figures 5.5 and 5.7 depicts two-dimensional and contour representations for this case respectively. These results were obtained from simulations performed with the filter method.

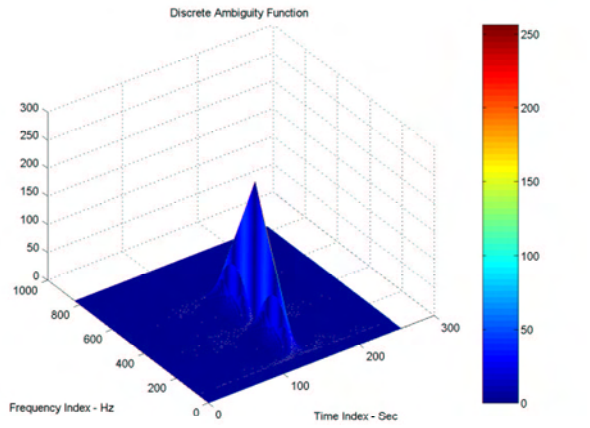

Figure 5.2: Three-dimensional Representation of the Ambiguity Function for a Transmitted Pulse Signal

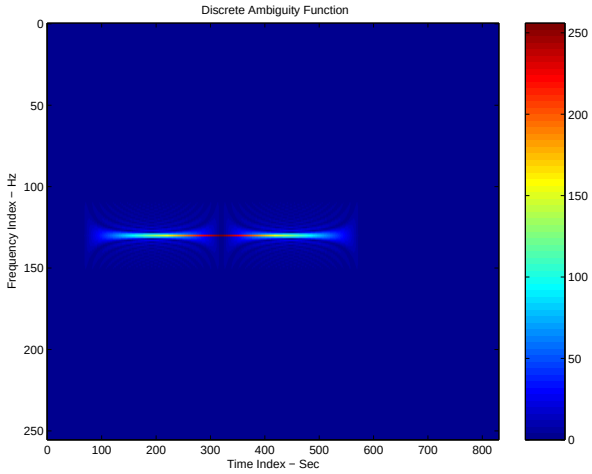

Figure 5.3: Two-dimensional Representation of the Ambiguity Function for a Transmitted Pulse Signal

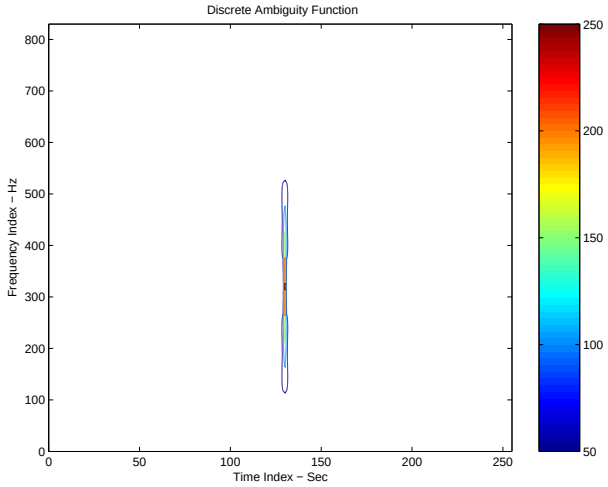

Figure 5.4: Contour Plot of the Ambiguity Function for a Transmitted Pulse Signal

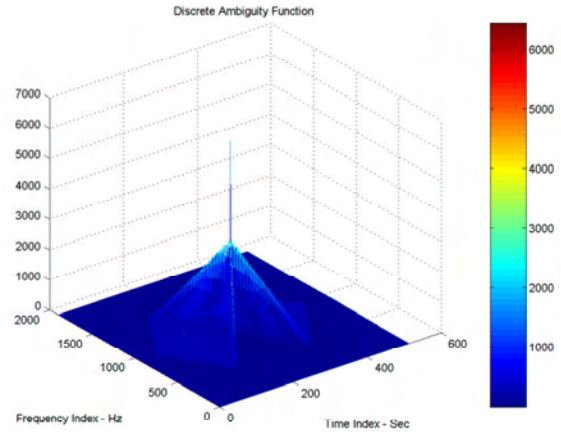

Figure 5.5: Three-dimensional Representation of the Ambiguity Function for a Transmitted Chirp Signal

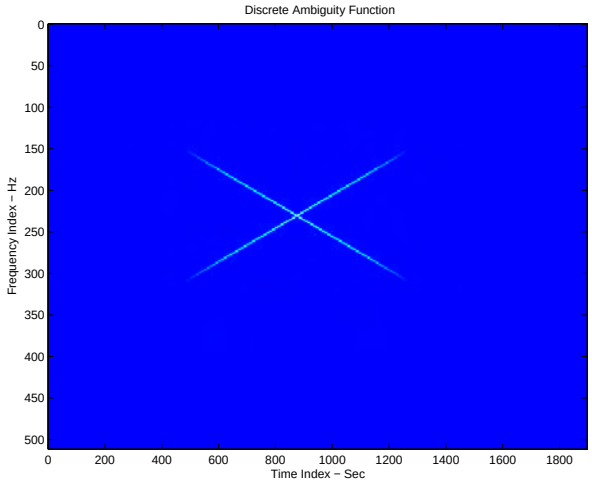

Figure 5.6: Two-dimensional Representation of the Ambiguity Function for a Transmitted Chirp Signal

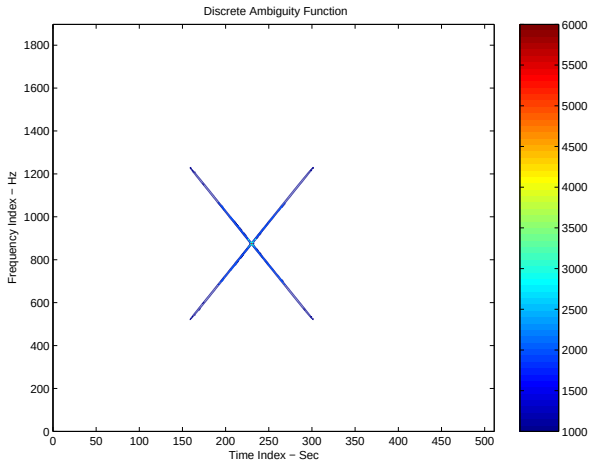

Figure 5.7: Contour Plot of the Ambiguity Function for a Transmitted Chirp Signal

5.2.2 Comparison between Linear and Cyclic DFT Methods

Another significant result is the cyclic formulation of the discrete ambiguity function in terms of signal algebra operators. An example is presented as follows in order to make a comparison between the results obtained from linear and cyclic DFT methods simulated with the parameters shown in table 5.2.

Table 5.2: Simulation Parameters for DFT Method

Signal Type	Chirp
Transmitted Signal Length (N)	32
Time Delay (sec)	40
Frequency Shift (Hz)	20

Figure 5.8 displays a two-dimensional representation of the ambiguity function for a transmitted chirp signal of length $N = 32$. This result was obtained from the linear DFT method which implements a linear cross-correlation operation. The resulting surface has a dimension of $L \times N$, where $L = N + M - 1 = 32 + 72 - 1 = 103$, i.e., the ambiguity function was computed as a rectangle or strip. Notice that the maximum is located at the time-delay and frequency-shift parameters of the received signal.

Now, consider the result depicted in figure 5.9. It was obtained from the cyclic DFT method which implements a cyclic cross-correlation operation. Then, the image displayed is a matrix of dimension $L \times L$, i.e., the ambiguity function was computed as a square. Observe that the ambiguity function appears three times along the frequency index. Why this happens? This modularity occurs due to the cyclic property introduced by the family of functions or characters presented in the expression of the ambiguity function. These characters are given by W_L^{kn}, where $W_L = e^{-j\frac{2\pi}{L}}$. From the fraction of the exponent it was encountered that $L/N = 103/32 \approx 3$, which is the number of computed ambiguity functions.

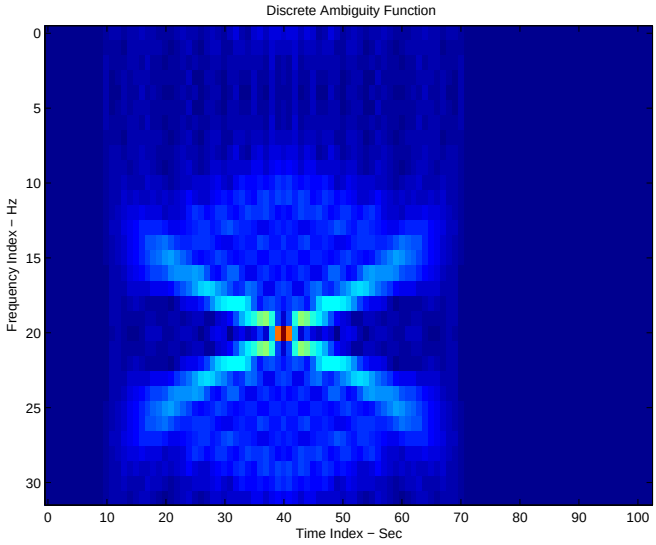

Figure 5.8: Chirp Ambiguity Function Computed by the Linear DFT Method

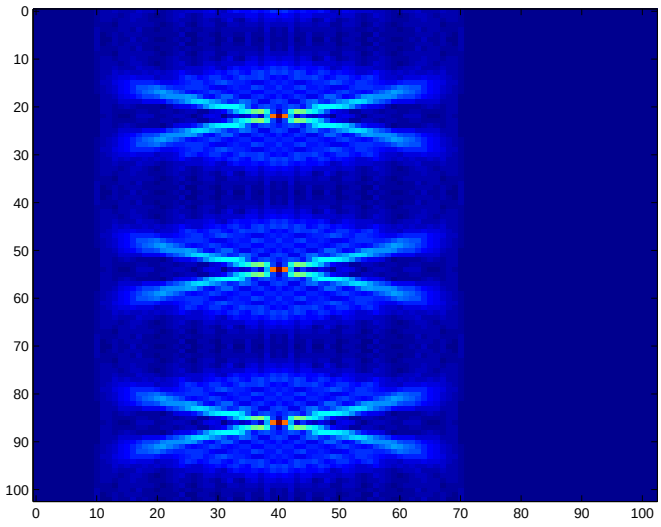

Figure 5.9: A Chirp Ambiguity Function Computed by the Cyclic DFT Method

5.3 Methodology Validation

The methodology used in this work to validate the proposed model is based in the theoretical formulations available in some literature and the results obtained by computing the ambiguity function directly from its mathematical expression [30]. Figure 5.10 presents the validation methodology employed in this thesis work.

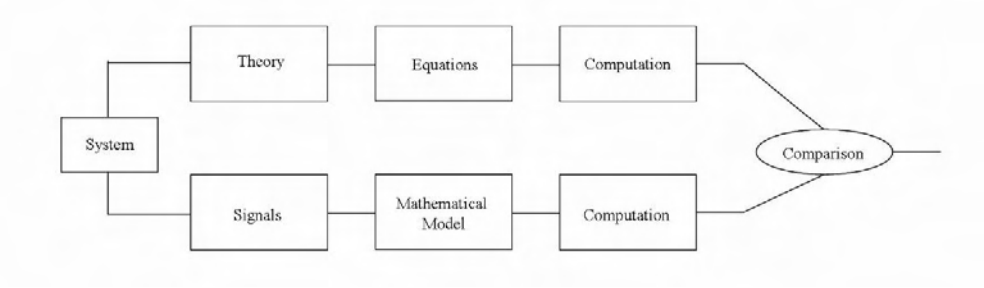

Figure 5.10: Validation Methodology for PSF simulations

Model validation was carried out by two specific cases of ambiguity function computation: for a transmitted pulse signal and for a transmitted pulse train. Figure 5.11 shows the methodology used to validate results on ambiguity function computed with the DFT method for a transmitted pulse signal.

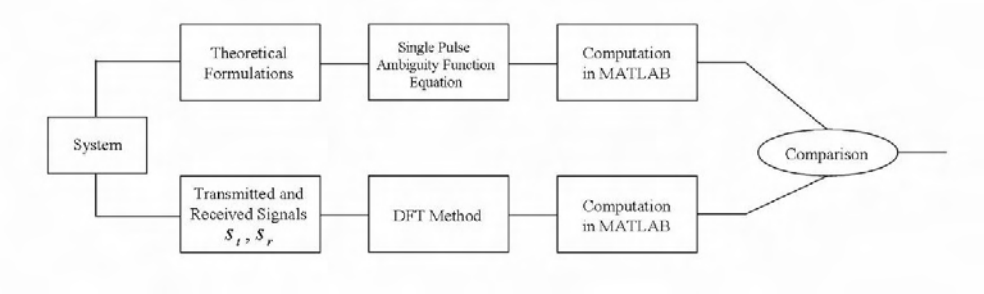

Figure 5.11: Validation Methodology for PSF simulations for a Transmitted Pulse Signal

The results of the DFT method were compared with the computations presented by Bassem [30]. These results were obtained from computing the ambiguity function of a single rectangular pulse as the direct computation of equation (4.25):

$$|A(\tau, f_d)| = \left| \left(1 - \frac{|\tau|}{\tau'}\right) \frac{\sin\left(\pi f_d \left(\tau' - |\tau|\right)\right)}{\pi f_d \left(\tau' - |\tau|\right)} \right| \tag{5.1}$$

Figure 5.12 presents the result of the direct computation of the ambiguity function of a single pulse signal. In figure 5.13 a result obtained with the DFT method for a single pulse ambiguity function is presented. According with the methodology, the model used to perform PSF simulations is valid. The results achieved by the DFT method are equivalent to those presented in the literature.

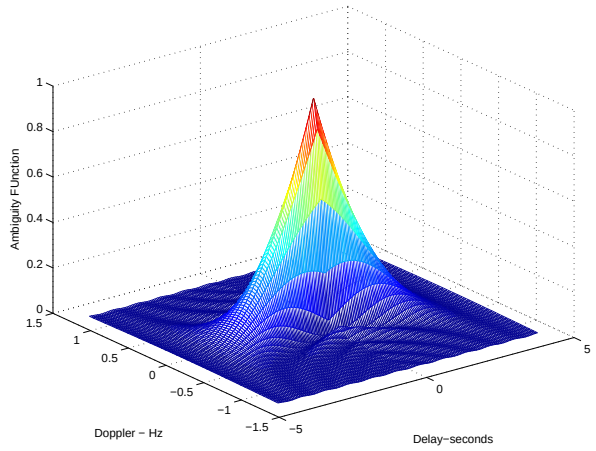

Figure 5.12: Single Pulse Ambiguity Function Computed directly from Equation

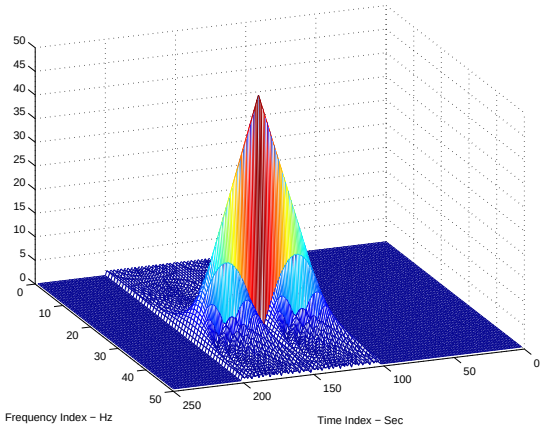

Figure 5.13: Single Pulse Ambiguity Function Computed with the DFT Method

In the case that the transmitted radar signal be a pulse train the validation methodology becomes the illustrated in figure 5.14.

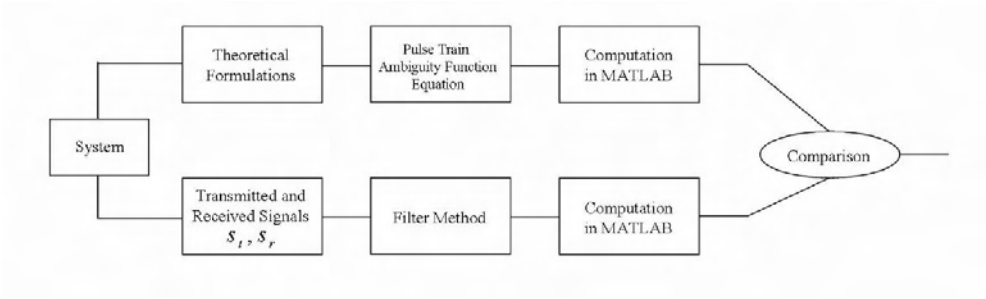

Figure 5.14: Validation Methodology for PSF simulations for a Transmitted Pulse Train

The coherent pulse train ambiguity function was computed with equation (4.29):

$$A(\tau, v) = \frac{1}{N} \sum_{q=-(N-1)}^{N-1} |A_1(\tau - qT, v)| \left| \frac{\sin[\pi v N - |q| T]}{\sin(\pi v T)} \right| ; \tau' < T/2 \qquad (5.2)$$

Where N is the number of pulses in the train, T is the pulse repetition interval (PRI), and τ' is the pulse width.

Figure 5.15 presents the result of the direct computation of the ambiguity function of a pulse train [30]. In figure 5.16 a result obtained with the filter method for a pulse train function is presented.

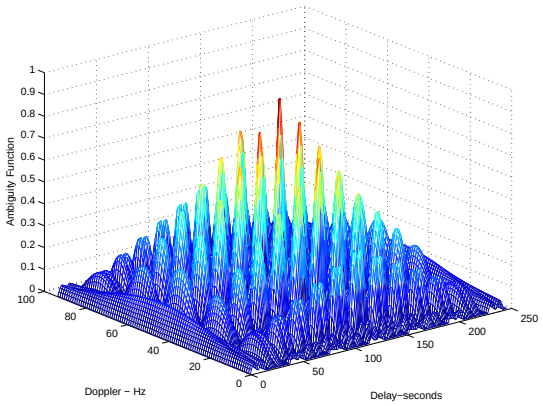

Figure 5.15: Pulse Train Ambiguity Function Computed directly from Equation

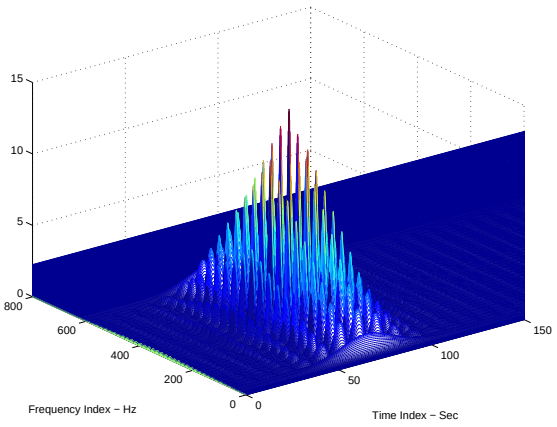

Figure 5.16: Pulse Train Ambiguity Function Computed with Filter Method

5.4 Future Work

The amounts of sensory data collected from SAR imaging systems receivers, are essentially, very large two-dimensional arrays of complex numbers (see figure 5.17). Therefore, the processing of this data is computationally expensive and it requires high memory and storage capacities. In this regard, large-scale PSFs simulations must be conducted in order to approximate them to the real SAR system processing problem. The term large-scale refers to big data sizes which demands scalability. However, the simulations carried out in Matlab encountered a memory limitation that limited scalability to a problem size smaller than SAR's.

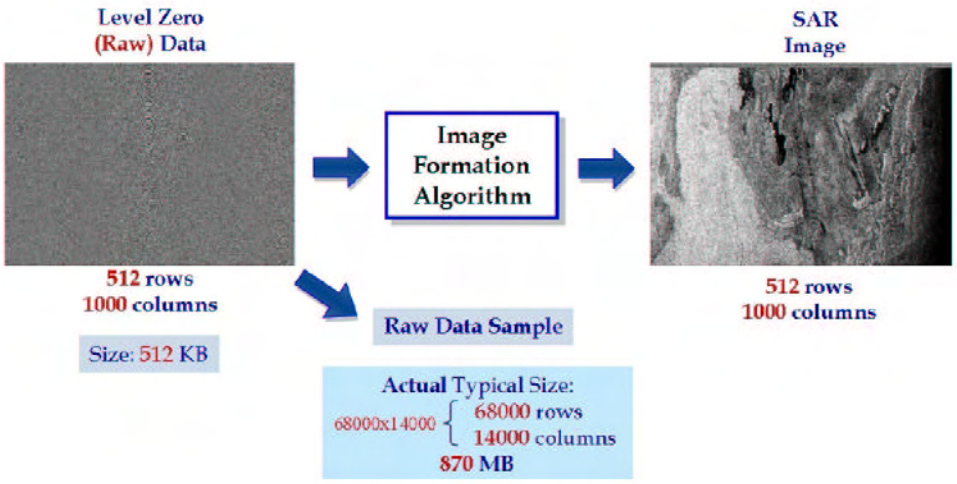

Figure 5.17: SAR Typical Data Size

For this reason, it is suggested that the SAR impulse response functions simulations be conducted on high-performance computers (HPC) with additional computational resources in terms of programming languages, memory, speed, and number of processors. Also, it is suggested that the SAR raw data generation model be executed on HPCs. Cur-

rently, the electrical and computer engineering department of the university of Puerto Rico at Mayaguez counts with a cluster system that could be used to perform these simulations.

The main system specifications are the following:

- IBM xSeries server, dual processors, 64 computing nodes.
- 130 MB cache Intel Pentium III (1.26 GHz and 512 KHz each).
- 63.8 GB of RAM.
- 1 terayte user disk space.
- Redhat v.7.3 operating system

Taking advantage of these computational resources, specially of the number of processors available, it is suggested to perform the simulations in a parallel mode. Figure 5.18 depicts a possible parallel approach for the implementation of the DFT method in which it can be parallelized either the Hadamard product operation or the FFTs.

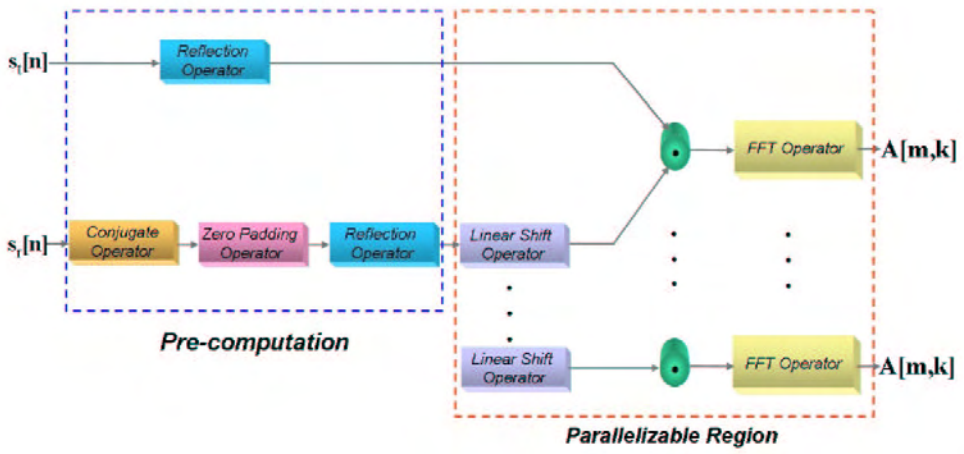

Figure 5.18: Parallel Approach for SAR Impulse Response Simulations

Some preliminary simulations results have been obtained using advanced architectures in preparation for future work on large scale modeling and simulation of SAR impulse response functions. All the results presented here were obtained from serial mode implementations. A library called FFTW was used to compute the FFTs operations presented in the DFT method algorithm. Appendix A describes how to program with FFTW.

The first simulations were performed on a symmetric multiprocessor system (SMP). This system has four Ultra SPARC II processors running at 400 MHz each, with 4 MB of local, high-speed external cache memory and Solaris v.8 (UNIX) operating system. The simulation algorithm was programmed in MATLAB, C and Fortran. Time results of these simulations are presented in table 5.3.

Table 5.3: Time Measurements of SAR PSFs Simulations on a SMP System

Image Size	Real Execution Time (sec)		User Time (sec)		System Time (sec)	
	C	FORTRAN	C	FORTRAN	C	FORTRAN
128×383	0.4740	0.2010	0.2200	0.1800	0.2600	0.0100
256×767	2.1830	0.6110	0.9000	0.5600	1.2700	0.0400
512×1535	8.0870	2.0530	3.4400	1.8300	4.6200	0.02100
1024×3071	33.2810	7.7270	13.7400	6.9800	19.4200	0.6200
2048×6143	134.174	30.6602	52.7300	27.3600	80.9900	2.7700
4096×12287	566.6400	131.8690	198.4800	111.5500	264.0000	12.8000
8192×24575	Segmentation Fault/Memory allocation error					

Figure 5.19 shows the run times comparison between the different language implementations of the DFT method. From this, notice that the FORTRAN implementation was the fastest. This is because FORTRAN has the necessary libraries to manage complex numbers. In addition, this language can adapt to the kind of architecture used, better than other languages.

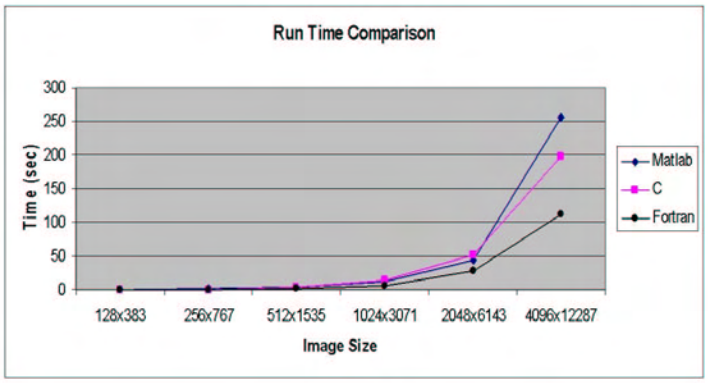

Figure 5.19: Run Time Comparison of SMP Simulations

Finally, the simulations were performed on the IBM cluster in serial mode using C and FORTRAN languages. The time results are presented in table 5.4.

Table 5.4: Time Measurements of SAR PSFs Simulations on the IBM Cluster

Image Size	Real Execution Time (sec)		User Time (sec)		System Time (sec)	
	C	FORTRAN	C	FORTRAN	C	FORTRAN
128×383	0.0330	0.0310	0.0300	0.0200	0.0100	0.0100
256×767	0.1130	0.0990	0.0900	0.0700	0.0200	0.000
512×1535	0.3960	0.3570	0.3600	0.2900	0.0300	0.0700
1024×3071	1.4890	1.4420	1.3200	1.1600	0.1700	0.2800
2048×6143	8.5210	6.8780	5.0100	5.8400	0.7600	0.8500
4096×12287	KILLED					

CHAPTER 6

Conclusions

The conclusions of this work are presented and resumed in this chapter as follows:

- A discrete-time, discrete-frequency computational model for SAR system raw data generation was presented, which was used to study the characteristics of Point Spread Functions (PSF), in a time-frequency context, through the efficient computation of the discrete ambiguity function.
- The SAR point spread function was modeled and simulated as the discrete ambiguity function in a time-frequency context.
- A large scale algorithm for Point Spread Functions (PSF) computation using the DFT method has been implemented in Fortran and C languages.
- The PSF simulations have been performed on a Symmetric Multiprocessor System (SMP) and a cluster architecture in serial mode using computational methods.
- Two methods for computing the ambiguity function were implemented: the DFT method, and the filter method.
- The DFT method is the more computationally efficient algorithm to simulate the ambiguity function.

BIBLIOGRAPHY

[1] J. Curlander and R. MacDonough. *Synthetic Aperture Radar: Systems and Signal Processing*. Jhon Wiley and Sons, Inc., 1991.

[2] D. Rodríguez. *Computational Signal Processing and Sensor Array Signal Algebra: A Representation Development Approach*. First book draft, September 2002.

[3] National Communications System Technology and Standards Division. Telecommunications: Glossary of telecommunications terms. http://www.its.bldrdoc.gov/fs-1037/, August 1996.

[4] C. Elliot and M. Hektner. Wetland resources of yellowstone national park. http://wetlands.fws.gov/Pubs_Reports/Yellowstone/Yell_pdfs/intro.pdf.

[5] R. E. Blahut. Theory of remote surveillance algorithms. *Radar and Sonar, Part I*, Vol. 32, New York, 1991.

[6] L. Auslander and R. Tolimieri. Characterizing the radar ambiguity functions. *IEEE Transactions on Information Theory*, 30:832–836, November 1984.

[7] R. Tolimieri and S. Winograd. Computing the ambiguity surface. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33:1239–1245, October 1985.

[8] L. Auslander and R. Tolimieri. Computing decimated finite cross-ambiguity functions. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36:359–364, March 1988.

[9] D. Rodríguez, J. Seguel, and E. Cruz. Algebraic methods for the analysis and design of time-frequency signal processing algorithms. *ISCAS Conference Proceedings*, 1:196–199, May 1993.

[10] M. Richman, T. Parks, and R. Shenoy. Discrete-time, discrete-frequency, time-frequency representations. *International Conference on Acoustics, Speech and Signal Processing (ICASP'95)*, 2:1029 –1032, May 1995.

[11] M. Richman, T. Parks, and R. Shenoy. Discrete-time, discrete-frequency, time-frequency analysis. *IEEE Transactions on Signal Processing*, 46:1517 –1527, June 1998.

[12] A. Ozdemir and O. Arikan. Fast computation of the ambiguity function and the wigner distribution on arbitrary line segments. *IEEE Transactions on Signal Processing*, 49:381–393, 2001.

[13] H. Ozaktas, O. Arikan, M. Kutay, and G. Bozdagi. Digital computation of the fractional fourier transform. *IEEE Transactions on Signal Processing*, 44:2141–2150, September 1996.

[14] E. Mozeson and N. Levanon. Matlab code for plotting ambiguity functions. *IEEE Transactions on Aerospace and Electronic Systems*, 38:1064 –1068, July 2002.

[15] G. Franceschetti, M. Migliaccio, and D. Riccio. The SAR simulation: An overview. *IEEE International Geoscience and Remote Sensing Symposium*, 1995.

[16] G. Franceschetti and R. Lanari. *Synthetic Aperture Radar Processing*. CRC Press, December 1998.

[17] G. Verdone, R. Viggiano, L. Candela, and V. Giannini. Processing algorithms for COSMO-SkyMed SAR sensor. *IEEE International Geoscience and Remote Sensing Symposium, IGARSS '02*, 5:2771 –2774, 2002.

[18] G. Franceschetti, A. Iodice, D. Riccio, and G. Ruello. A 2-d fourier domain approach for spotlight SAR raw data simulation of extended scenes. *IEEE International Geoscience and Remote Sensing Symposium and the 24th Canadian Symposium on Remote Sensing*, June 24-28 2002.

[19] G. Franceschetti, M. Migliaccio, D. Riccio, and G. Schirinzi. Saras: A synthetic aperture radar (SAR) raw signal simulator. *IEEE Transactions on Geoscience and Remote Sensing*, 30:110–123, January 1992.

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

[21] S. Lang. *ALGEBRA*. Addison-Wesley Publishing Company, third edition, 1993.

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

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

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

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

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

[27] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Signal Processing Series, 1989.

[28] W. J. Karplus, G. A. Bekey, and B. Y. Kogan. *Modeling and Simulation: Theory and Practice, A Memorial Volume for Professor Walter J. Karplus (1927-2001)*. Kluwer Academic Publishers, 2003.

[29] P. A. Fishwick. Computer simulation. *IEEE Potentials*, 15:24–27, Feb.-March 1996.

[30] B. R. Mahafza. *Radar Systems Analysis and Design using MATLAB*. ChapmanHall/CRC, 2000.

[31] D. Rodríguez. SAR point spread signals and Earth surface property characteristics (invited paper). *SPIE's 44th Annual Meeting and Exhibition, Denver, Colorado*, July 1999.

[32] V. Nallur. Functional programming: the basics. http://www.ncst.ernet.in/education/apgdst/pgpfac/slides.shtml.

[33] M. Frigo and S. Jhonson. FFTW: An adaptive software architecture for the FFT. *ICASSP Conference Proceedings*, 1998.

[34] M. Frigo and S. Johnson. The fastest fourier transform in the west. *MIT-LCS-TR-728*, September 1997.

APPENDICES

APPENDIX A

Programming with FFTW

FFTW or the faster Fourier transform in the west is a publicly available C package of subroutines that computes one-dimensional and multidimensional discrete Fourier transforms (DFT). FFTW can compute transforms of real and complex data of arbitrary input sizes not restricted to power of two. These subroutines were developed at the Massachusetts Institute of Technology (MIT) by Matteo Frigo and Steven G. Johnson. FFTW implements the Cooley-Tukey fast Fourier transform algorithm for serial and parallel environments. Parallel implementations of FFTW are also available for shared memory systems using threads and for distributed-memory systems using the message passing interface (MPI) [33].

The main feature of FFTW is its ability to adapt the computation to the hardware architecture in order to achieve best performance. This is possible due to the internal structure of FFTW. The computation of the transform is carried out by three fundamental operation modules described as follows:

1. *FFTW Planner:* The FFTW planner uses a dynamic programming algorithm to create a special data structure called a *plan* [34]. This structure consists of a efficcient combination of specialized blocks of C code called *codelets*. Each codelet computes part of the transform. The planner evaluates different combinations of codelets and its execution time, selecting the fastest plan. FFTW provides the function *fftw_create_plan,* for interactions with the users. Through this function users can create a plan for a

specific transform. Once the plan is created it can be used as many times as needed. When the plan is used for all FFTs computations it must be destroyed calling the function *fftw_destroy_plan(plan)*.

2. *Codelet Generator:* The key of the performance of FFTW reside in the automatic generation of codelets that computes transforms of small sizes. The generator itself implements the Cooley-Tukey algorithm using the dive-and-conquer approach to recursively compute the FFT. This generator behaves like a special-purpose compiler that produces long blocks of optimized code, which are easy to modify with simple changes[34].

   The codelet generator was written in the functional programming language Caml Light from the ML family, which is free available and highly portable. Caml capabilities were exploited by FFTW for computing the Cooley-Tukey algorithm using symbolic arithmetic. This symbolic arithmetic is then simplified by many transformations in a simplification phase to obtain the optimized blocks of C code.

3. *FFTW Executor:* Finally, the FFT computation is performed by a C function called the executor, which accepts as input the plan generated by the FFTW planner and the data array to be transformed. The plan provides to the executor a sequence of intructions that specifies the type of transform that will be computed. Then, the executor function varies according to the transform that is desired to compute.

Figure A.1 summarizes the interaction of the user, the planner, the codelet generator and the executor in order to compute the FFT. As it can be viewed in this figure the users interact with FFTW through the planner and the executor. They create the plan using the function *fftw_create_plan* which accepts as inputs the length of the transform, the direction of the transform (forward or barckward), and other optional flags. Then, using another function that accepts as inputs the plan and the data, FFTW executes the plan to obtain the resulting transform.

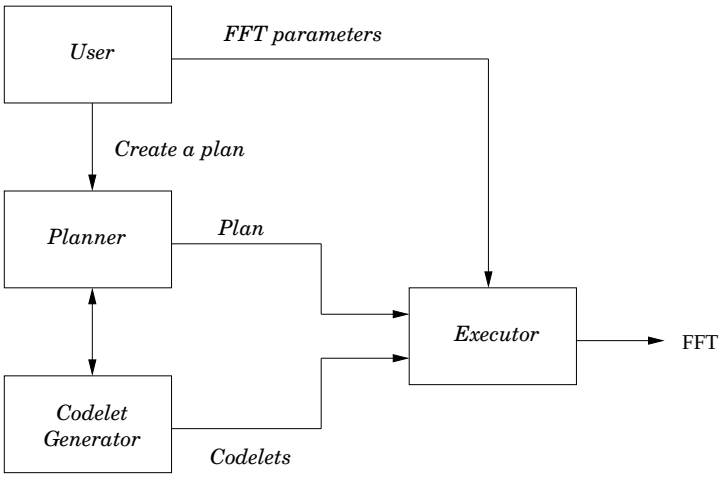

Figure A.1: FFTW Computational Scheme

Algorithm A.1 shows the basic usage of the FFTW functions in C to compute one-dimensional FFTs of complex data in serial mode. In this case, the function to execute the plan for a complex one-dimensional FFT is *fftw_one*. This function varies according with the dimension of the transform.

Algorithm A.1

```
#include <fftw.h>
 ...
{
 fftw_complex  *in, *out;
 fftw_plan plan;
 ...
```

```
 // Create a specific plan
 plan=fftw_create_plan(N, FFT_DIR, flags);
 ...
 // Execute the plan
 fftw_one(plan, in, out);
 ...
 fftw_destroy_plan(plan);
}
```

In the example above, the first argument, N, in the function *fftw_create_plan* corresponds to the size of the transform and must be any non-negative integer, the second argument can be either *FFTW_FORWARD* or *FFTW_BACKWARD* which computes forward and inverse FFT respectively. The last argument can be either of two flags *FFTW_MEASURE* or *FFTW_ESTIMATE*. *FFTW_MEASURE* is used to measure the execution time of several FFTs in order to find the best plan to compute the transform. On the other hand, *FFTW_ESTIMATE* constructs a reasonable plan that may be sub-optimal for the FFT computation.

Algorithms A.2 and A.3 present the codes for compute the one-dimensional and the two-dimensional FFTs respectively in C. Both, one-dimensional and multi-dimensional FFTs must be compiled as follows in a Solaris environment with the GNU C compiler: gcc file.c -lfftw -lm -o executable_file

Algorithm A.2

```
#include <fftw.h>
#include <stdio.h>
 int N;
 int main(int argc,char* argv[])
{
 fftw_complex *in, *out;
 fftw_plan p;
 int i;
 in = ( fftw_complex *) malloc(N*sizeof(fftw_complex));
 out = ( fftw_complex *) malloc(N*sizeof(fftw_complex));
 p=fftw_create_plan(N,FFTW_FORWARD,FFTW_ESTIMATE);
 fftw_one(p,in,out);
 fftw_destroy_plan(p);
 return 0;
```

```
}
```

Algorithm A.3

```
#include <fftw.h>
#include <math.h>
#include <stdio.h>
 int N, M;
 int main(int argc,char* argv[])
{
 fftw_complex *in;
 fftwnd_plan p;
 in = ( fftw_complex *) malloc(N*M*sizeof(fftw_complex));
 p=fftw2d_create_plan(N,M,FFTW_FORWARD,
                      FFTW_MEASURE|FFTW_IN_PLACE);
 fftwnd_one(p,in,NULL);
 fftwnd_destroy_plan(p);
 return 0;
}
```

In the multidimensional case is possible to have an additional flag such as *FFTW_IN_PLACE*, in which the output data overwrite the input data. Thus, it requires half as much memory as the out-of-place transform (the default). The arguments N and M correspond to the number of rows and the number of columns respectively.

The following two algorithms present FORTRAN codes for one and two-dimensional transforms respectively. The parameters used in the functions that creates the plans are the same described for the C examples. Compile these routines using f77 compiler as follows: f77 file.F -lfftw -lm -o executable_file

Algorithm A.4

```
      program fftw1D
      implicit none
   #include "fftw_f77.i"
      integer N
      parameter(N=32)
      double complex in
      dimension in(N)
      integer*8 plan
      call fftw_f77_create_plan(plan,N,M,FFTW_FORWARD,
      +                          FFTW_ESTIMATE + FFTW_IN_PLACE)
      call fftw_f77_one(plan,in,0)
      call fftw_f77_destroy_plan(plan)
```

```
      end
```

Algorithm A.5

```
      program fftw2D
      implicit none
   #include "fftw_f77.i"
      integer N, M
      parameter(N=32)
      parameter(M=64)
      double complex in
      dimension in(N,M)
      integer*8 plan
      call fftw2d_f77_create_plan(plan,N,M,FFTW_FORWARD,
      +                           FFTW_ESTIMATE + FFTW_IN_PLACE)
      call fftwnd_f77_one(plan,in,0)
      call fftwnd_f77_destroy_plan(plan)
      end
```

The available functions to compute multidimensional FFTs using FFTW routines provide for parallel implementations on distributed-memory machines supporting MPI. In this case, the transform data is distributed over multiple processes, so that each process gets only a portion of the array. In algorithm A.6 the basic C code to compute complex two-dimensional FFTs using MPI FFTW is presented. To compile this use: mpicc file.c -lfftw_mpi -lfftw -lm -o executable_file

Algorithm A.6

```
#include <fftw_mpi.h>
#include <math.h>
#include <stdio.h>
 int Nx, Ny;
 int main(int argc,char* argv[])
{
 fftw_complex *in;
 fftwnd_mpi_plan p;
 MPI_Init(&argc,&argv);
 in = ( fftw_complex *) malloc(Nx*Ny*sizeof(fftw_complex));
 p = fftw2d_mpi_create_plan(MPI_COMM_WORLD,Nx,Ny,FFTW_FORWARD,
                                  FFTW_ESTIMATE);
 fftwnd_mpi(p,1,in,NULL,FFTW_NORMAL_ORDER);
 fftwnd_mpi_destroy_plan(p);
 MPI_Finalize();
```

```
 return 0;
}
```

Algorithm A.7 presents the basic C code to compute complex two-dimensional FFTs using the parallel FFTW routines for shared-memory threads on SMP systems. These routines take extra parameters such as the number of threads to use. Compile this routine as follows: gcc file -lfftw_threads -lfftw -lm -lpthread -o executable_file

Algorithm A.7

```
#include <fftw_threads.h>
#include <math.h>
#include <stdio.h>
 int N, M, NT;
 int main(int argc,char* argv[])
{
 fftw_complex *in;
 fftwnd_plan p;
 int T;
 T = fftw_threads_init();
 in = ( fftw_complex *) malloc(N*M*sizeof(fftw_complex));
 p = fftw2d_create_plan(N,M,FFTW_FORWARD,
                        FFTW_MEASURE|FFTW_IN_PLACE);
 fftwnd_threads_one(NT,p,in,NULL);
 fftwnd_destroy_plan(p);
 return 0;
}
```

APPENDIX B

MATLAB Functions

This appendix contains the MATLAB functions developed for the SAR point spread function modeling and simulation. These functions are presented as follows:

1. *directcomp.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  LINEAR AMBIGUITY FUNCTION: Direct Computation
%
%  This function accepts as input two radar signals:
%  transmitted (st) and received (sr) signals.
%  It returns the discrete ambiguity function
%  from the linear cross-correlation of these signals
%  computed in direct form.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = directcomp(st,sr)

lst=length(st);
lsr=length(sr);

srcon=conj(sr);
L=lst+lsr-1;
M = L + lsr;

srcp = zeropaddingop(srcon,M);

A=zeros(L,lst);

for m=0:lsr-1
    for k=0:lst-1
        for n=0:lst-1
```

```
        h = st(n+1)*srcp(n+m+1);
        A(m+1,k+1)=A(m+1,k+1) + h*exp(-j*2*pi*k*n/lst);
       end
    end
end

A = abs(A);
```

2. *cycdirectcomp.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  CYCLIC AMBIGUITY FUNCTION: Direct Computation
%
%  This function accepts as input two radar signals:
%  transmitted (st) and received (sr) signals.
%  It returns the discrete ambiguity function
%  from the cyclic cross-correlation of these signals
%  computed in direct form.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = cycdirectcomp(st,sr)

lst=length(st);
lsr=length(sr);

srcon=conj(sr);
L=lst+lsr-1;
srcp = zeropaddingop(srcon,L);
stp = zeropaddingop(st,L);
B=zeros(L,L);

for m=0:L-1
    for k=0:L-1
        for n=0:L-1
        g =srcp(mod(n+m,L)+1);
        t = stp(n+1);
        h = g.*t;
        B(m+1,k+1)=B(m+1,k+1) + h*exp(-j*2*pi*k*n/lst);
       end
    end
end

A = abs(B)
```

3. *DFTmethod.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  DFT METHOD FUNCTION
%
%  Implemented Method: Linear Transform (DFT) Method
%  This function accepts as input two radar signals:
%  transmitted (st) and received (sr) signals.
%  It returns the discrete ambiguity function
%  from the linear cross-correlation of these signals
%  using the DFT method.
%  Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = DFTmethod(st,sr)

lst = length(st);
lsr = length(sr);
L=lst+lsr-1;

src=conj(sr);
srp = zeropaddingop(src,L);
sri=indexrevop(srp,L);
sti=indexrevop(st,lst);

A=zeros(L,lst);

for shift=0:1:lsr-1
   sris=linleftshift(sri,lst,shift);
   z=sti.*sris;
   A(lsr-shift,:)=fft(z);
end

G = abs(A);
```

4. *cycDFTmethod.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CYCLIC DFT METHOD FUNCTION
%
% Implemented Method: Cyclic Transform (DFT) Method
% This function accepts as input two radar signals:
% transmitted (st) and received (sr) signals.
% It returns the discrete ambiguity function
% from the cyclic cross-correlation of these signals
% Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = cycDFTmethod(st,sr)

lst = length(st);
lsr = length(sr);
L=lst+lsr-1;

src=conj(sr);
stp = zeropaddingop(st,L);
srp = zeropaddingop(src,L);

A=zeros(L,L);
stp=stp(1:L);

for shift=0:L-1
    sris=cycleftshift(srp,L,shift);
    z=stp.*sris;
    A(shift+1,:)=fft(z);
end

A = abs(A);
```

5. *filtermethod.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  FILTER METHOD FUNCTION
%
%  Implemented Method: Filter Method
%  This function accepts as input two radar signals:
%  transmitted (st) and received (sr) signals.
%  It returns the discrete ambiguity function
%  from the autocorrelation of these signals
%  using the Filter method.
%  Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A = filtermethod(st,sr)
lst = length(st);
lsr = length(sr);
L=lst+lsr-1;

src=conj(sr);

srp = zeropaddingop(src,L);
sri=indexrevop(srp,L);
Fsri=fft(sri);

z =zeros(1,L);

for k=0:lst-1
   for n=0:lst-1
   z(n+1)=st(n+1).*exp(sqrt(-1)*2*pi.*n.*k/lst);
   end
     Z=fft(z);
     Gfsz=Fsri.*Z;
     Grev=ifft(Gfsz);
     Gc=indexrevop(Grev,L);
     G(:,k+1)=Gc';
end

A=abs(G(1:lsr,:));
```

6. *indexrevop.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INDEX REVERSAL OPERATOR (Reflection Operator)
%
% This function accepts two parameters as input:
% A signal (or vector) and its length.
% This function returns the index reversed version
% of the input signal
% Programmed by: Hilaura Nava.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function reversedsignal = indexrevop(signal,N)

if length(signal) ~= N
   error('Second input parameter does not correspond ...
          to the true signal length')
   else

   for k=0:N-1;
   reversedsignal(k+1)=signal(N-k);
   end
end
```

7. *zeropaddingop.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ZERO PADDING OPERATOR
%
% Zero pad program for 1-d signals
% This function accepts two parameters as input:
% A 1-d signal (or vector) and the desired maximum
% length that will have the input signal
% This function returns the zero padded version
% of the input signal
% Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function paddedsignal = zeropaddingop(signal, maxlength)

if length(signal) > maxlength
    error('Signal length is larger than desired maximum length')
else
    padlength = maxlength - length(signal);
    paddedsignal = [signal, zeros(1,padlength)];
end
```

8. *linleftshift.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  LINEAR LEFT SHIFT OPERATOR
%
%  This function accepts three parameters as input:
%  A signal (or vector) x, the length of the Hadamard product
%  operation M, and the number of shifts.
%  This function returns the same input vector but
%  shifted to the left, from a shift zero to N-1
%  Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function shiftedsignal = linleftshift(x,N,shift)

shiftedsignal=zeros(1,N);

for i=0:N-1
    shiftedsignal(i+1)=x(i+shift+1);
end
```

9. *cindexrevop.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CYLCIC INDEX REVERSAL OPERATOR
%
% This function accepts two parameters as input:
% A signal (or vector) and its length
% This function returns the index reversed version
% of the input signal
% Programmed by: Hilaura Nava.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function reversedsignal = cindexrevop(signal,N)

reversedsignal(1)=signal(1);

if length(signal) ~= N
    error('Second input parameter does not ...
           correspond to the true signal length')
    else

    for i=0:N-2
    reversedsignal(i+2)=signal(N-i);
    end
end
```

10. *cycleftshift.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  CYCLIC LEFT SHIFT OPERATOR
%
%  This function accepts three parameters as input:
%  A signal (or vector) x, and the number of shifts.
%  This function returns the same input vector but
%  shifted to the left, from shift zero to N-1
%  Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function shiftedsignal = cycleftshift(x,L,shift)

shiftedsignal=zeros(1,L);

for i=0:L-1
  shiftedsignal(i+1)=x(mod(i+shift,L)+1);
end
```

11. *cycrightshift.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  CYCLIC RIGHT SHIFT OPERATOR
%
%  This function accepts three parameters as input:
%  A signal (or vector) x, the x signal length L,
%  and the number of shifts.
%  This function returns the same input vector but
%  shifted to the right, from shift zero to N-1
%  Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function shiftedsignal = cycrightshift(x,L,shift)

shiftedsignal=zeros(1,L);

for i=0:L-1
    shiftedsignal(i+1)=x(mod(i-shift,L)+1);
end
```

12. *signals.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SIGNAL GENERATION FUNCTION
%
% This function returns the transmitted and received
% radar signals. Its input consists of:
%  * Signal parameters:
%      type: type of signal (pulse, cos, chirp)
%      leng: signal length
%      delay: time delay of teh received signal
%      shif: frequency shift of the received signal
% Programmed by: Hilaura Nava.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [st,sr]=signals(type,leng,delay,shif,Fs)

if type==1
%--- Single Pulse Signal ----%
     st=[ones(1,leng) ];
     sr=[zeros(1,delay) st];
     shift=exp(j*2*pi.*(0:length(sr)-1)*(shif)/length(st));
     sr=sr.*shift;
 elseif type==2
 %--- Cosine Signal ---%
     st=[cos(2*pi.*(1:leng)/leng)];
     sr=[zeros(1,delay) st];
     shift=exp(j*2*pi.*(0:length(sr)-1)*(shif)/length(sr));
     sr=sr.*shift;

 elseif type==3
%--- Chirp Signal---%
     Fs=1000;
     Ts=1/Fs;
     fc=100;
     tinc=Ts;
     Tm=100*Ts;
     m=leng/100;
     V=m*Tm;
     N=V*Fs;
     t=-V/2:tinc:V/2-Ts;
     finc=Fs/N;
     f=-(Fs/2):finc:(Fs/2)-finc;
     M=5;
     a=2*pi*100/V;
     b=2*pi*fc;;
     c=0.25*pi;
```

```
%*********************************************************%
     st=M*cos(1*a*(t.*t)+b*t+c);
     sr=[zeros(1,delay) st];
     shift=exp(j*2*pi.*(0:length(sr)-1)*(shif)/length(st));
     sr=sr.*shift;
end
```

13. *readComplex.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% READ COMPLEX FUNCTION
%
% This function reads a sequence of complex numbers
% from a file. The data contained in the file must
% have the following format:
% Ex: 1.0000 0.80000
%     0.2300 1.21000
% Two columns, separated by one space. The first
% column correspond to the real part of the number,
% and the second column to the imaginary part.
% Both numbers are floating-point numbers.
% Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = readComplex(fname)

fid = fopen(fname, 'r');
if fid == -1
    error(['Could not open file : ' fname ' for reading.'])
end
cnt = 1;
sizeofV = 500;
inc = 100;
R = zeros(sizeofV,1);
I = R;
while(~feof(fid))
  line = fgetl(fid);
  tmp = sscanf(line, '%f', 2);
  R(cnt) = tmp(1);
  I(cnt) = tmp(2);
  cnt = cnt+1;
  if cnt > sizeofV
      R(sizeofV + inc) = 0;
      I(sizeofV + inc) = 0;
    end
end
```

```
out = complex(R(1:cnt-1),I(1:cnt-1));
out=transpose(out);
```

14. *writecomplex.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WRITE COMPLEX FUNCTION
%
% This function writes the real and imaginary parts of a complex
% vector (from the Matlab workspace) in two files separately in
% fixed-point notation and 8 decimals.
% This function accepts three parameters:
% input: a complex vector located in the Matlab Workspace
% fname1: the name of the output file where the real part of
% the vector will be stored
% fname2: the name of the output file where the imaginary part
% of the vector will be stored
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function writecomplex(input, fname1, fname2)

realPart = real(input);
imagPart = imag(input);

fid = fopen(fname1,'W');
fid1 = fopen(fname2,'W');

for i=1:length(realPart)
    fprintf(fid, '%.8f\r\n', realPart(i));
    fprintf(fid1,'%.8f\r\n', imagPart(i));
end

fclose(fid);
fclose(fid1);
```

15. *writecomplex2.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WRITE COMPLEX FUNCTION 2
%
% This function writes a complex vector from the Matlab
% workspace to a file in fixed-point notation and 8 decimals.
% The function accepts two parameters:
% input: a complex vector located in the Matlab Workspace
% fname: the name of the output file where the vector will be
% stored
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function writecomplex(input, fname)

realPart = real(input);
imagPart = imag(input);

fid = fopen(fname,'W');

for i=1:length(realPart)
    fprintf(fid, '%.8f %.8f\r\n', realPart(i), imagPart(i));
end

fclose(fid);
```

16. *complexsig.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COMPLEXSIG FUNCTION
%
% This function accepts as input the name of the
% files that contains the real part and the imaginary
% part of the transmitted and received radar signals.
% fname1 : File name of the transmitted signal real part
% fname2 : File name of the transmitted signal imaginary part
% fname3 : File name of the received signal real part
% fname4 : File name of the received signal imaginary part
% The function returns to MATLAB workspace the transmitted and
% received signals in complex format.
% Programmed by: Hilaura Nava
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [st,sr]= complexsig(fname1, fname2, fname3, fname4)

   rst = load(fname1,'-ascii');
   ist = load(fname2,'-ascii');
   rsr = load(fname3,'-ascii');
   isr = load(fname4,'-ascii');

   st = complex(rst,ist);
   sr = complex(rsr,isr);

   st=transpose(st);
   sr=transpose(sr);
```

17. *writeptoh.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WRITE FILE .h FUNCTION
%
% This function writes a complex vector from the Matlab
% workspace to a file in fixed-point notation and 8 decimals.
% The function accepts two parameters:
% input: a complex vector located in the Matlab Workspace
% fname: the name of the output file where the vector will be
% stored
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function writeptoh(input, fname, N)

realPart = real(input);
imagPart = imag(input);

fid = fopen(fname,'W');
```

```
fprintf(fid,'%s%d%s','float datare[',N,']={');
for i=1:N-1
    fprintf(fid, '%.8f,', realPart(i));
end
fprintf(fid, '%.8f', realPart(N));
fprintf(fid,'%s\n','};');

fprintf(fid,'%s%d%s','float dataim[',N,']={');
for i=1:N-1
    fprintf(fid, '%.8f,', imagPart(i));
end
fprintf(fid, '%.8f', imagPart(N));
fprintf(fid,'%s\n','};');

fclose(fid);
```

APPENDIX C

C Programs

This appendix contains the C programs and functions developed for SAR point spread function modeling and simulation.

1. *amblindftm.c*

```
/*************************************************************
This program computes the Ambiguity Function through the
Linear DFT Method. In this program we use the FFTW libraries

  To compile use: gcc amblindftm.c -lfftw -lm -o exec_file
  To execute: exec_file data1.txt N M
  data1.txt: contains transmitted and received signals(st,sr)
  N: is the transmitted signal length
  M: is the received signal length
*************************************************************/

#include <fftw.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

int Readsig(char *,int, int, int, fftw_complex *, fftw_complex *);
int N, M, K;
char input1[20];

int main(int argc,char* argv[])
{
  fftw_complex *st;
  fftw_complex *sr;
  fftw_complex *in, *out;
```

```
  fftw_plan p;

  FILE *output;
  int i, j, k;

  N = atoi(argv[2]);
  M = atoi(argv[3]);
  K=N+M-1;
  fftw_complex **Matrix;


    in =   ( fftw_complex *) malloc(N*sizeof(fftw_complex));
    out =   ( fftw_complex *) malloc(N*sizeof(fftw_complex));

    st = ( fftw_complex *) malloc(K*sizeof(fftw_complex));
    if (st==NULL)
       printf("Error/n");

    sr = ( fftw_complex *) malloc(K*sizeof(fftw_complex));

/*Zero padding using memset function
    sr = (fftw_complex*) memset(sr,0,K*sizeof(fftw_complex));
    if (sr==NULL)
       printf("Error/n");

   Matrix = ( fftw_complex **) malloc(M*sizeof(fftw_complex*));
   for(k=0;k<M;k++){
       Matrix[k]=(fftw_complex *)malloc(N*sizeof(fftw_complex));
   }

  strcpy(input1,argv[1]);
  if(Readsig(input1, N, M, K, sr, st)==-1)
       return -1;

  p=fftw_create_plan(N,FFTW_FORWARD,FFTW_ESTIMATE);

/***********Shift & Hadamard & FFT*************/
  for(i=0;i<M;i++){
    for(j=0;j<N;j++){
      in[j].re=(st[j].re*sr[j+i].re) - (st[j].im*sr[j+i].im);
      in[j].im=(st[j].re*sr[j+i].im) + (st[j].im*sr[j+i].re);

    }
    fftw_one(p,in,out);
```

```
       for(k=0;k<N;k++){

        fftw_one(p,in,out);
         for(k=0;k<N;k++){
         Matrix[M-i-1][k].re = out[k].re;
         Matrix[M-i-1][k].im = out[k].im;
  }
}
   fftw_destroy_plan(p);

  free(st);
  free(sr);
  free(srz);
  free(in);
  free(out);

  return 0;
 }
```

2. *amblindftm2.c*

```
/************************************************************
This program computes the Ambiguity Function through the
Linear DFT Method. In this program we use the FFTW libraries

  To compile use: gcc amblindftm2.c -lfftw -lm -o exec_file
  To execute: exec_file data1.txt data2.txt N
  data1.txt: contains transmitted signal (st)
  data2.txt: contains received signal (sr)
  N: is the transmitted signal length
In this program the received signal has its minimum
length, i.e, 2N
The result of thsi program is a Matrix, which is the
discrete ambiguity function
************************************************************/
#include <fftw.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

int readVector(fftw_complex[],int,int,char[]);

int N, M, K;

char input1[20];
```

```
char input2[20];

int main(int argc,char* argv[])
{
  fftw_complex *st;
  fftw_complex *sr;
  fftw_complex *sm, *in, *out;
  fftw_plan p;

  FILE *output;
  int i, j, k;

  N = atoi(argv[2]);
  M=(2*N)-1;          //Minimun Received Signal Length
  K=N+M-1;
  fftw_complex **Matrix;


    in =   ( fftw_complex *) malloc(N*sizeof(fftw_complex));
    out =   ( fftw_complex *) malloc(N*sizeof(fftw_complex));

    st = ( fftw_complex *) malloc(K*sizeof(fftw_complex));
    if (st==NULL)
    printf("Error/n");


    sr = ( fftw_complex *) malloc(K*sizeof(fftw_complex));
    sr=(fftw_complex*) memset(sr,0,K*sizeof(fftw_complex));
    if (sr==NULL)
    printf("Error/n");

 sm = ( fftw_complex *) malloc(N*sizeof(fftw_complex));

    Matrix = ( fftw_complex **) malloc(M*sizeof(fftw_complex*));
  printf("Separando memoria\n");
    for(k=0;k<M;k++)
       Matrix[k]=(fftw_complex *)malloc(N*sizeof(fftw_complex));


  strcpy(input1,argv[1]);
  strcpy(input2,argv[3]);
printf("Leyendo file %s \n",input1);
  if(readVector(st,N,N,input1)==-1)
       return -1;
printf("Leyendo file %s \n",input2);
```

```
  if(readVector(sr,M,K,input2)==-1)
       return -1;

/* ReadFiles(sr, st, N, K, M, input1, input2);*/

/***********Shift & Hadamard************/
for(i=0;i<M;i++){
    for(j=0;j<N;j++){
      Matrix[M-i-1][j].re=(st[j].re*sr[j+i].re)-(st[j].im*sr[j+i].im);
      Matrix[M-i-1][j].im=(st[j].re*sr[j+i].im)+(st[j].im*sr[j+i].re);

    }
}

/************* FFTW ***********************/
fflush(stdout);
   for(i=0;i<M;i++){
      for(j=0;j<N;j++){
         in[j]=Matrix[i][j];
     }
      p=fftw_create_plan(N,FFTW_FORWARD,FFTW_ESTIMATE);
      fftw_one(p,in,out);

      for(k=0;k<N;k++){
         Matrix[i][k]=out[k];
   }
}

fftw_destroy_plan(p);

  return 0;
 }
```

3. *I/O Functions*

```
/*************************************************************/
 This routine reads st and sr from a file and perform
 index-reversal of sr
/*************************************************************/
int Readsig(char *filename, int N, int M, int K,
            fftw_complex *sr, fftw_complex *st){
    FILE *FIDinputfile;
    int i;
    if( (FIDinputfile = fopen(filename, "r")) == NULL){
       printf("File could not be opened\n");
       return -1;}
```

```
    else {
        for(i=N-1;i>=0;i--){
           if(feof(FIDinputfile)){
              printf("Error EOF found\n");
              fclose(FIDinputfile);
              return -1;
    }

    fscanf(FIDinputfile, "%lf %lf %lf %lf",&sr[i+(K-N)].re,
          &sr[i+(K-N)].im,&st[i].re,&st[i].im);
    }
     for(i=K-N-1;i>=K-M;i--)
       fscanf(FIDinputfile, "%lf %lf", &sr[i].re,&sr[i].im );
       fclose(FIDinputfile);
       }
       return 0;
}
/*************************************************************/
 This program writes the 2-d output of the linear DFT method
 algorithm in a file
/*************************************************************/
if((output=fopen("OUT.txt", "w")) == NULL)
printf("File could not be opened for write operation\n");
  else{
       for(j=0; j<M; j++){
         for(i=0; i<N; i++){
          fprintf(output,"%lf\t%lf\n", Matrix[j+M*i].re,
                  Matrix[j+M*i].im, "\n");
                         }
                           }
         fclose(output);
  }
/*************************************************************/
```

4. *createdata.c*

```
/*************************************************************/
 This routine creates a file "datain.txt" with complex, double
-precision, and floating-point random numbers. This data is
used in the ambiguity fuction simulations
/*************************************************************/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main() {
```

```
  int i;
  FILE *outfile;

  if ( (outfile = fopen("datain.txt","w")) == NULL) {
       printf("Error\n");
       return -1;
  }
  for(i =0;i<1024;i++){
       fprintf(outfile,"%lf %lf\n",drand48(),drand48());
  }
  fclose(outfile);
return 0;
}
```

5. *cretedata.c*

```
/*************************************************************/
 This routine creates a file "data.txt" with complex, double
-precision, and floating-point random numbers. This data
is stored in teh file data.txt in the FORTRAN complex format.
This data is to be read by a FORTRAN program.
/*************************************************************/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main() {

  int i;
  FILE *outfile;

  if ( (outfile = fopen("data.txt","w")) == NULL) {
        printf("Error\n");
        return -1;
  }
  for(i =0;i<1024;i++){
        fprintf(outfile,"(%lf,%lf)\n",drand48(),drand48());
  }
  fclose(outfile);
return 0;
}
```

APPENDIX D

FORTRAN Programs

This appendix contains the FORTRAN programs and functions developed for discret ambiguity function modeling and simulation.

1. *DAFsubru.F*

```
c-------------------------------------------------------------c
  Program DAF
c This program reads the transmitted (St) and received (Sr) c
c signals from a file and compute the Discrete Ambiguity    c
c Each operator were programmed as a separate FORTRAN       c
c subroutine.                                               c
c Function  by means of the Linear DFT Method               c
c  The parameters of this program are:                      c
c   LT = transmitted signal length                          c
c   LR = received signal length                             c
c   LZ = output length                                      c
c Compile this program as follows:                          c
c  f77 DAFsubru.F -lfftw -lm -o exec_file                   c
c-------------------------------------------------------------c
      Parameter(LT=8)

      Parameter(LR=2*LT-1)
      Parameter(LZ=LT+LR-1)

      Complex*16 St(LT),Sr(LR)
c-------------------------------------------------------------c
      Open(unit=27,file='transig2',status='old')
      Do j = 1 ,LT
             Read(27,fmt=*,end=99)St(j)
c             print *,St(j)
      EndDo
```

```
99    close(unit=27)
c-------------------------------------------------------------c
      Open(unit=27,file='recsig2',status='old')
      Do j = 1 ,LR
             Read(27,fmt=*,end=100)Sr(j)
             Sr(j)=conjg(Sr(j))
c       print *,Sr(j)
      EndDo
100   close(unit=27)
      Call Zeropad(St,Sr,LT,LR,LZ)
      END

c---------------ZERO PADDING-----------------c
      SUBROUTINE Zeropad(St,Sr,LT,LR,LZ)
      Complex*16 Sr(LR),Srz(LZ)

      Do 15 j=1,LR
            Srz(j)=Sr(j)
15    CONTINUE
      Do 18 k=LR+1,LZ
            Srz(k)=(0,0)
18    CONTINUE
      Do 30 i=1,LZ
c     print *,Srz(i)
30    CONTINUE
 Call IndexRev(St,LT,Srz,LR,LZ)
      RETURN
      END
c-------------INDEX REVERSAL---------------c
      SUBROUTINE IndexRev(St,LT,Srz,LR,LZ)
      Complex*16 St(LT),Srz(LZ)
      Complex*16 Sti(LT),Sri(LZ)

      Do 40 k=0,LT-1
           Sti(k+1)=St(LT-k)
c     print *,Sti(k+1)
40    CONTINUE
      Do 50 j=0,LZ-1
           Sri(j+1)=Srz(LZ-j)
c     print *,Sri(j+1)
50    CONTINUE
      Call Shift(Sti,Sri,LT,LR)
      RETURN
      END
```

```
c-------------Shift & Hadamard-------------c
      SUBROUTINE Shift(Sti,Sri,LT,LR)
      Complex*16 Sti(LT),Sri(LT+LR-1)
      Complex*16 G(LT),B(LR,LT)

      Do 60 i=0,LR-1
      Do 70 j=0,LT-1
            G(j+1)=Sti(j+1)*Sri(j+i+1)
70    CONTINUE
  Do 80 j=0,LT-1
      B(LR-i,j+1)=G(j+1)
80    CONTINUE
60    CONTINUE
      Call FFT(B,LT,LR)
      RETURN
      END

c---------------- FFTW --------------------c
      SUBROUTINE FFT(B,LT,LR)
      Complex*16 A(LR,LT),B(LR,LT)

#include "fftw_f77.i"

      Integer N
      N=LT
      double complex in, out
      dimension in(N),out(N)

      integer i

      integer plan

      Do 10 i=1,LR
      Do 20 j=1,LT
      in(j)=B(i,j)
20    CONTINUE
      call fftw_f77_create_plan(plan,N,FFTW_FORWARD,FFTW_ESTIMATE)
      call fftw_f77(plan,1,in,1,0,out,1,0)
      Do 30 k=1,LT
      A(i,k)=out(k)
30    CONTINUE
10    CONTINUE
      Open(unit=27,file='AFmatrix',status='new')
      Do 40 j = 1,LT
      Do 50 i = 1,LR
```

```
            Write(27,fmt=*)A(i,j)
50      CONTINUE
40      CONTINUE

        call fftw_f77_destroy_plan(plan)
        RETURN
        END
```