

**SIMULATION OF MILLIMETER WAVE RADAR
RETURN FROM A THREE DIMENSIONAL
ENVIRONMENTAL SCENE**

by

PHILIPPE BERISSET

Ingénieur en Electronique Aéronautique
Ecole Nationale Supérieure de l'Aéronautique et de l'Espace (FRANCE), 1992

Submitted to the Department of Aeronautics and Astronautics
in Partial Fulfillment of the Requirements for the
Degree of

MASTER OF SCIENCE
in Aeronautics of Astronautics

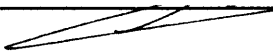
at the


MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1993

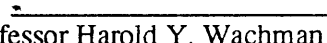
© Philippe Bérisset, 1993. All rights reserved.

The author hereby grants to MIT permission to reproduce and to
distribute publicly copies of this thesis document in whole or in part.

Signature of the Author  Department of Aeronautics and Astronautics
May 7, 1993

Certified by  Dr. Ying-Ching E. Yang
Thesis Supervisor
Research Scientist, Research Laboratory of Electronics

Certified by  Professor Jin A. Kong
Thesis Supervisor
Department of Electrical Engineering and Computer Science

Accepted by  Professor Harold Y. Wachman
Chairman, Department Graduate Committee



JUN 08 1993

LIBRARIES

SIMULATION OF MILLIMETER WAVE RADAR RETURN FROM A THREE DIMENSIONAL ENVIRONMENTAL SCENE

by

PHILIPPE BERISSET

Submitted to the Department of Aeronautics and Astronautics
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Aeronautics of Astronautics

ABSTRACT

A millimeter wave stepped frequency radar image simulation process applied to three dimensional earth terrain is developed. This radar image simulation involves four domains of study: terrain volume and surface backscattering properties modeling, radar sensor modeling, radar image simulation techniques, and application of these methods to computerized simulation.

Two theoretical scattering models are used to characterize the terrain. Volume scattering properties are modeled with a two-layer anisotropic random medium model and surface scattering is calculated using a small perturbation method.

A Gaussian statistics model is applied to the scattered pulses from the illuminated terrain to simulate the speckle pattern that appear on real radar images.

Two simulation techniques are investigated. In the first simulation method, the synthetic high range resolution stepped frequency system is considered as the basis for idealized image generation. The second image synthesis approach models the real sensor and involves a Fourier transform based high resolution image recovery algorithm. Since the imaged area is not flat, both of the simulation approaches take into account the shadowing effect.

From these techniques, two computer programs are developed and implemented. The simulation results from the idealized method and the real sensor approach are compared together and to the actual terrain map.

Thesis Supervisor: Dr. Jin Au Kong.

Title: Professor of Electrical Engineering and Computer Science.

Thesis Supervisor: Dr. Ying-Ching Eric Yang.

Title: Research Scientist, Research Laboratory of Electronics.

Acknowledgment

First of all, I would like to thank the Délégation Générale pour l'Armement of the French Department of Defense for its continuous support during this year.

I would like to express my deepest appreciation to Professor Jin Au Kong for his guidance in my finding the subject of this study.

I would like to thank Dr. Ying-Ching Eric Yang for his helpful advice, guidance and concern throughout every stage of my research. This thesis would never have materialized without his fruitful criticisms and suggestions. I am also deeply grateful to him for reading my thesis.

Many thanks are due to Li-Fang Wang and Ante Salcedo who contributed to the collection of backscattering coefficients. I would also like to thank my colleagues Joel Johnson, Ali Tassoudji, Hong Tat Ewe, Pierre Coutu, Chih-Chien Hsu and John Oates for enlightening discussions.

I would like to express my profound gratitude to my fiancée Nathalie for her patience, support and encouragement.

My fondest thanks to my parents for their lifelong faith in me. I credit all my accomplishment to their unwavering support.

Table of Contents

Abstract	2
Acknowledgment	3
List of Figures	7
List of Tables	10
Chapter 1	INTRODUCTION 11
1.1	Background 11
1.2	Review of Previous Work 14
1.2.1	Review of Terrain Scattering Models 14
1.2.2	Radar Images 15
1.3	Description of the Thesis 17
Chapter 2	RADAR SCATTERING MODEL 19
2.1	Introduction 19
2.2	Two-Layer Anisotropic Random Medium Model 22
2.3	Small Perturbation Method 23
2.4	Computation of the Return Power 24
2.4.1	Coherent Summation of the Returned Pulses 24
2.4.2	Computation of the Return Power 30

Chapter 3	SIMULATION TECHNIQUES	33
3.1	Introduction	33
3.2	Simulation Techniques	35
3.2.1	Idealized Image Synthesis	35
3.2.2	Real Sensor Image Simulation	38
3.3	Simulation Program Architecture	45
Chapter 4	PRE-PROCESSING OF THE INPUT DATA	47
4.1	Introduction	47
4.2	Pre-Processing Method	49
4.3	Mathematical Formulation	53
4.3.1	Definitions	53
4.3.2	Computation of the Characteristics of the Small Cells	54
Chapter 5	COMPUTATION OF THE SIMULATED RADAR IMAGE	58
5.1	Introduction	58
5.2	Method, Algorithms	59
5.2.1	Introduction, Definitions	59
5.2.2	Algorithms	61
Chapter 6	COMPUTERIZED IMAGE SYNTHESIS RESULTS, INTERPRETATION	66
6.1	Introduction	66
6.2	Image Synthesis: Flat Terrain	69
6.2.1	Idealized Approach	70
6.2.2	Real Sensor Approach	71
6.2.3	Interpretation	72
6.3	Image Synthesis: Vegetation	73
6.3.1	Idealized Approach	74

6.3.2	Interpretation	75
6.3.3	Real Sensor Approach	77
6.3.4	Interpretation	77
6.4	Image Synthesis: Road, Vegetation, and Lake	78
6.4.1	Idealized Approach	79
6.4.2	Real Sensor Approach	80
6.4.3	Interpretation	80
6.5	Image Synthesis: Vegetation and Runway	81
6.5.1	Idealized Approach	81
6.5.2	Real Sensor Approach	82
6.5.3	Interpretation	82
6.6	Limitations	83
6.6.1	Image Distortion	83
6.6.2	Scattering Model Limitations	84
6.6.3	Shadowing Effect Approximation	85
Chapter 7	SUMMARY, CONCLUSION,		
	AND SUGGESTED FUTURE WORK	87
8.1	Summary and Conclusions	87
8.2	Suggested Future Work	89
APPENDIX A		90
	THREE DIMENSIONAL GEOMETRY DERIVATION		
APPENDIX B		92
	C CODE FOR IDEALIZED RADAR IMAGE SYNTHESIS		
APPENDIX C		113
	C CODE MODIFICATIONS FOR REAL SENSOR IMAGE SYNTHESIS		
Bibliography		119

List of Figures

Figure 1.1	Typical radar imaging system.	12
Figure 1.2	Types of data presentations.	16
Figure 2.1	Simulated image from a flat grass field.	20
Figure 2.2	Two-layer random medium model.	22
Figure 2.3	Randomly rough surface.	23
Figure 2.4	Evaluation of the phase correction term Δr_n	29
Figure 3.1	Range width of the idealized radar beam.	35
Figure 3.2	(a) First scanning simulated beam. (b) One step after (a). (c) When the right border is reached, the beam starts from the left again..	36
Figure 3.3	Actigram for the conversion processing.	40
Figure 3.4	Image distortion.	41
Figure 3.5	Actigram for the radar image synthesis system.	45
Figure 3.6	Actigram for radar image synthesis.	46
Figure 4.1	The total azimuthal width of the area to be scanned is divided into radial lines.	49
Figure 4.2	Definition of the horizontal projection of the small cells.	50
Figure 4.3	Definition of the non-flat small cells.	50
Figure 4.4	The program determines in which cell the current point is.	50
Figure 4.5	Definition of the two points of intersection I_1 and I_2 of the vertical plane V and the four sides of the terrain cell.	51

Figure 4.6	(a) and (b) The shadowing parameter of the cell take three values.	52
Figure 4.7	Geometrical definitions.	53
Figure 4.8	Definition of the four points of intersection between the vertical plane V and the four lines including the sides of the terrain cell.	54
Figure 4.9	Projection on the horizontal plane. Discrimination criterion used to find I_1 and I_2	55
Figure 4.10	Definition of the terrain type of the small cells.	56
Figure 4.11	Case where the shadowing parameter is equal to one.	57
Figure 5.1	The illuminated area is approximated by a rectangle.	59
Figure 5.2	The radar return corresponds to the terrain between the two wave fronts.	61
Figure 5.3	Scan of the terrain and computation of the return power.	62
Figure 5.4	Summation of the electric fields, idealized approach.	63
Figure 5.5	Computation of the power, idealized approach.	63
Figure 5.6	Summation of the electric fields, real sensor approach.	64
Figure 5.7	Computation of the power, real sensor approach.	65
Figure 6.1	Elevation data map. Scanned region for vegetation only: bottom area. Scanned region for road, vegetation and lake: top area. Scanned region for vegetation and runway: middle area.	68
Figure 6.2	Scanned area for flat terrain simulation.	69
Figure 6.3	Synthesized image for flat terrain, idealized sensor simulation.	70
Figure 6.4	Synthesized image for flat terrain, real sensor simulation.	71
Figure 6.5	Scanned region, non-flat terrain, vegetation only.	73
Figure 6.6	Synthesized image for non-flat terrain simulation (oversampling factor set to 1).	74

Figure 6.7	Synthesized image for non-flat terrain simulation (oversampling factor set to 2). 75
Figure 6.8	Elevation map corresponding to the imaged region. 76
Figure 6.9	Synthesized image for non-flat terrain real sensor simulation (vegetation). 77
Figure 6.10	Scanned region, non-flat terrain (road, vegetation, and lake). 78
Figure 6.11	Synthesized image for non-flat terrain idealized simulation (road, lake, and vegetation). 79
Figure 6.12	Synthesized image for non-flat terrain real sensor simulation (road, lake, and vegetation). 80
Figure 6.13	Synthesized image for non-flat terrain idealized simulation (vegetation and runway). 81
Figure 6.14	Synthesized image for non-flat terrain real sensor simulation (vegetation and runway). 82
Figure 6.15	Image distortion. 83
Figure 6.16	Backscattering coefficients are a function of the orientation α of the radar line of sight to the terrain. 84
Figure 6.17	The backscattering properties of trees depend on both α and β 84
Figure 6.18	a) Actual shadowing effect, the local maximum is taken into account b) The local maximum between two points is neglected. 86

List of Tables

Table 3.1	Conversion processing.	40
Table 4.1	Characteristics computed for each cell.	50
Table 4.2	Definitions.	53
Table 5.1	Definitions.	60
Table 6.1	Terrain types, characteristics, and corresponding gray level.	67
Table 6.2	Flat terrain radar image simulation characteristics.	70
Table 6.3	Non-flat terrain radar image simulation characteristics (vegetation area).	74
Table 6.4	Non-flat terrain radar image simulation characteristics (road, lake, and vegetation).	79
Table 6.5	Non-flat terrain radar image simulation characteristics (vegetation and runway).	81

Chapter 1

INTRODUCTION

1.1 Background

Even though the first reported activity in Millimeter Wave (MMW) technology began in the 1890s, the interest for this region of the electromagnetic spectrum has substantially increased only in the past twenty years (Curie, 1987). This phenomenon is mostly due to the recent advances in the technology of transmitters, receivers, and other components used in electromagnetic system applications. Therefore, the number of applications of MMW has also increased significantly.

The region of the electromagnetic spectrum ranging from 30 to 300 GHz (i.e. wavelengths between 1 cm and 1 mm) is usually referred to as the MMW region, although the IEEE Standard (IEEE, 1976) defines millimeter waves as the region between 40 and 300 GHz excluding the widely used region around 35 GHz.

Millimeter Wave Radar (MMWR) systems have some advantages compared to conventional radar which use microwave frequencies. Actually, one characteristic of the MMW frequency range is that for a given physical antenna size the beamwidth is smaller and the antenna gain is higher than at microwave frequencies (Curie, 1987). This characteristic is interesting for applications where the size and the weight are major concerns such as for missile seekers. Furthermore, MMWR systems offer a higher angular tracking accuracy than microwave radar systems.

Another characteristic of MMWR is that the atmospheric attenuations due to clouds, rain and fog are typically very low. The superiority of MMWR over optical and infrared systems for penetration of clouds, fog and in general adverse environments is also one of the reasons of the recent development of this research field.

Since the remote sensing of the earth's surface has many important applications such as topographical mapping, terrain determination and characterization, and terrain-sensing guidance, an all-weather capability is sometimes required. MMWR imaging systems are useful in performing remote sensing in these special conditions. The issue is that, in the past, radar imagery has lacked the high-resolution of optical imagery systems. However, techniques have been developed to substantially improve the radar resolution, widening the field of its applications.

Figure 1.1 shows a typical radar imaging system. The terrain is illuminated by a radar system. Pulses of millimeter wave energy are transmitted through the antenna within a narrow beamwidth. The illuminated terrain scatters the incident energy in all directions. The sensor receives and records the portion of this energy that is sent back in its direction. This return is then processed into an image. The purpose of this thesis is to study the prediction and simulation of such images and to develop an application program to simulate the radar image.

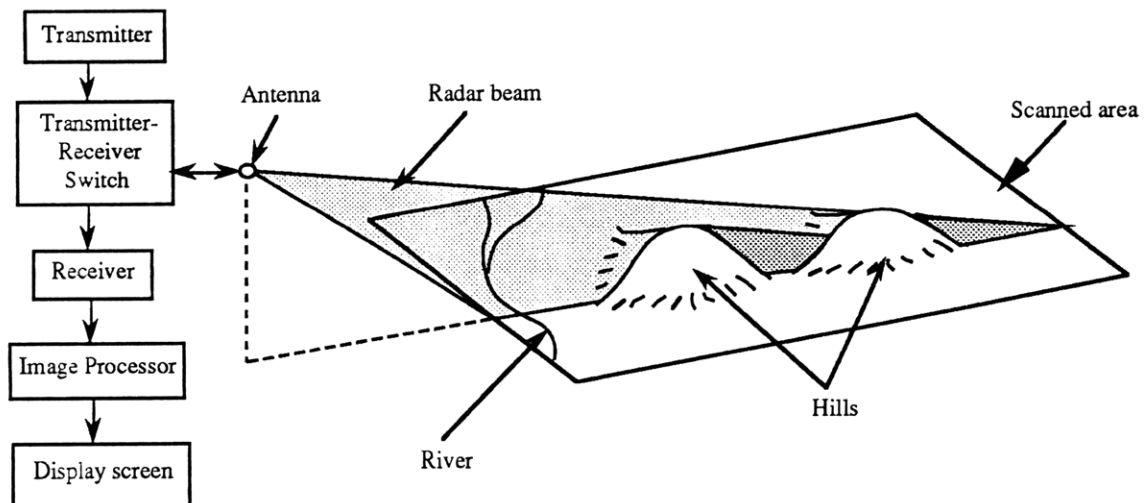


Figure 1.1: Typical radar imaging system.

There are several motivations for radar image prediction. First, since experiments are very expensive, it would be very useful to be able to simulate the result we would obtain with a radar system for which all the parameters are not fixed yet, before developing it. Secondly, for terrain-sensing guidance of aircraft, image comparison schemes could be used to update its navigation system. An on-board reference image could be compared with a prediction of the image that would be sensed if there were no guidance error to compute the position error. Another application would be terrain identification from its radar echo. In all cases, a good understanding of the backscattering characteristics will be necessary to provide accurate image predictions.

Since the scattering properties of arbitrary terrain have not been well understood in the past, predictions were always based on experimentally obtained data (Holtzman, 1978). The problem is that there is a quasi-infinite variety of terrain surface types. Thus, each time reflectivity coefficients for a given terrain were to be evaluated, this terrain had to be categorized into one of the terrain types for which experimental data had been measured. Obviously, extrapolations on the available data had to be made in order to estimate the characteristics of the given terrain for several orientation, polarizations and frequencies. Hence, this data evaluation method can not consistently provide accurate radar image prediction.

The prediction of the electromagnetic behavior of terrain surfaces was improved with the development of theoretical models which analytically describe the scattering properties of the Earth's surface. In this thesis, we will use two of these theoretical terrain surface models. The implementation of a radar image prediction procedure will require a good understanding not only of the backscattering characteristics of the terrain but also of image formation algorithms.

Architecture and development of image formation algorithms will be investigated in this thesis. Application programs corresponding to these algorithms will be implemented. Several synthesized images of real terrain will be analyzed.

1.2 Review of Previous Work

1.2.1 Review of Terrain Scattering Models

Since real terrain is a very complicated medium, a useful approach is to describe it in terms of the statistical properties of several of its parameters. Thus, we will use a model to describe how these characteristics vary within the terrain. Therefore, a very complicated terrain structure can be modeled in an equivalent fashion that is much easier to study.

As a general result, it was shown that the electromagnetic scattering properties of a terrain are due to both its surface roughness and the dielectric constant variations within the medium. However, since the contribution to the scattering properties of those two phenomena are usually not of the same order, two classes of models were developed. Volume scattering models account for the internal variations of the dielectric constant, whereas surface scattering models are based on the effects of the terrain roughness. In this thesis, we consider several types of terrain. Consequently, we will need both volume scattering and surface scattering models.

The volume scattering properties of terrain will be described using a two-layer anisotropic random medium model. In this model, a terrain is represented as a scattering region with an electrical permittivity which has a random part (this random part is assumed to be small compared to the fixed part). Stogryn (1974) first investigated this domain. Tsang and Kong (1976) developed a theory on this concept, in which they dealt with the problem of two-layer models. This theory was applied by Kong et al. (1979) to the modeling of the backscattered energies from several types of vegetation. They also matched the simulated data with experimental data.

The surface scattering model we will use was obtained by applying the small perturbation method to rough surfaces. The small perturbation method is due to Rice, 1963. Further investigation was done by Valenzuela (1968), who derived the scattered electromagnetic fields to the second order for a perfectly conducting and a dielectric surfaces. This method is only applicable when the surface variations are smaller than the incident wavelength and the slopes of the rough surface are relatively small.

These theoretical models allow the prediction of the backscattering properties for many different terrain types at many orientations, polarizations and frequencies.

For the past two decades, researchers at MIT have engaged in development of the random medium and rough surface scattering models. During the process, a series of programs, which incorporated both MIT's work and other prominent theoretical models, were written to deal with the applications to Earth remote sensing. This collection of programs became known as E.M.S.A.R.S. (Electromagnetic Models of Scattering Applied to Remote Sensing). In this thesis, the random medium volume scattering model and the small perturbation technique for rough surface scattering model will provide us with the backscattering coefficients for terrain features such as grass, trees, road, and lake.

1.2.2 Radar Images

Two major categories of radar image formation methods can be distinguished according to the processing employed (Huddleston, 1989): in-place imaging and object-motion imaging. In the first method, the object has a fixed attitude relative to the observer. The image is derived from several return observations (amplitude, polarization, and phase) of the object at one or more frequencies. This image is used to identify background and particular objects. The second method is based on the relative motion between the object and the observer. The image is then derived from a Doppler analysis of the return signal.

Radar displays are the link between radar information and a human operator. The display technique must be designed to improve the information transfer to the operator. Basically, there are two types of representations: the signal may be displayed as an orthogonal displacement (such as an oscilloscope) or as an intensity modulation of the screen (see next page, Figure 1.2). Most radar signals have the intensity-modulated type of display. A polar-coordinate indicator, where radar azimuth is presented as an angle and range as a radial distance, results in a maplike display (type P). In a cartesian-coordinate presentation, one may represent any combination of range, azimuth, height, elevation (types B, C, D, E, H). Several other types of radar image representation have been

developed due to the advances in digital signal processing. Instead of being displayed on cathode-ray tubes, images may be presented in many gray levels or even pseudo-colors on high resolution monitors.

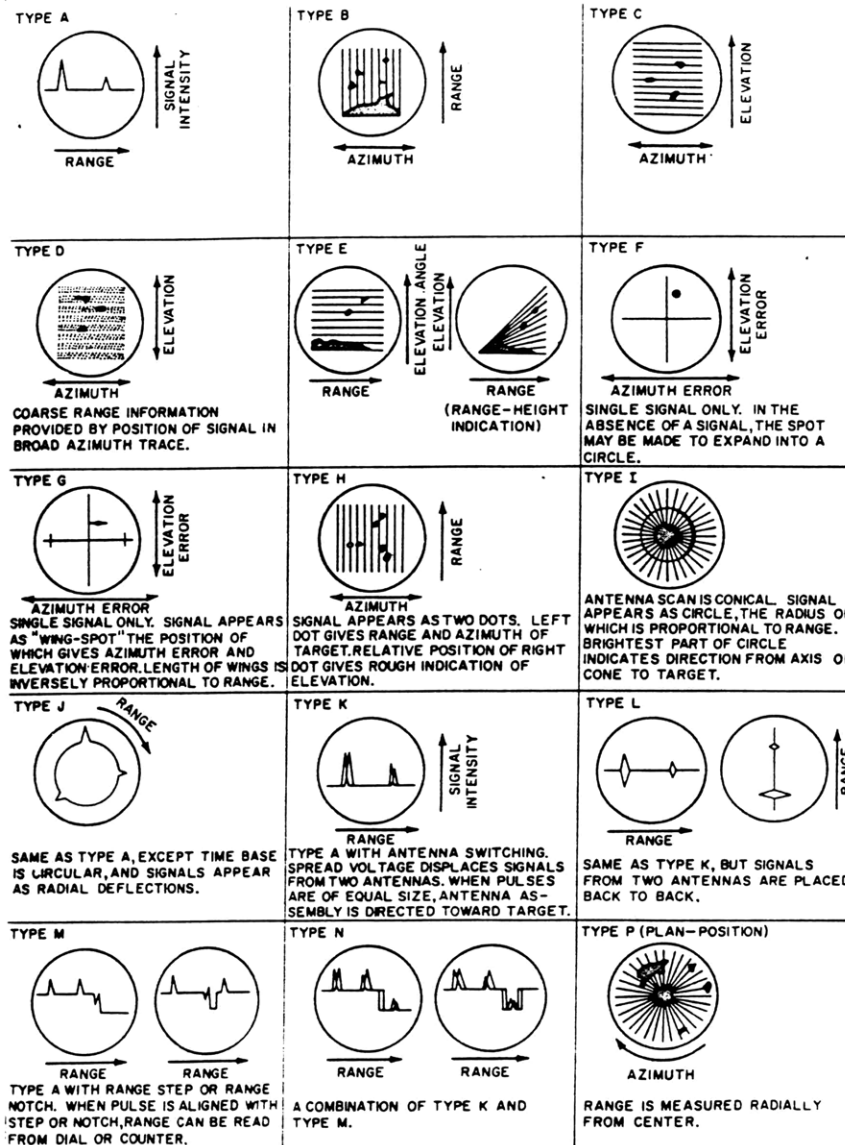


Figure 1.2: Types of data presentations. (From Reintjes and Coate, p.226, Fig. 6)

1.3 Description of the Thesis

Since the objective of this thesis is to deal with environmental scenes, we will only use in-place imaging. This thesis deals with the methods that can be applied to the simulation of the radar return image of a three dimensional environmental scene. We will first study an idealized image synthesis approach and then develop a method to take care of the real sensor case. The eventual objective of this study is to provide a computerized radar image simulator based on the investigated methods.

In Chapter 2, the radar scattering model that will be used all throughout this study is introduced. The analytical computation of the coherent summation of all the return wave fronts originating from the scatterers of the observed terrain is derived for a fully polarimetric analysis. The backscattering properties of these scatterers are obtained using the two theoretical scattering models also introduced in this chapter. From now on, it will always be considered that the studied radar has only the HH polarization. The analytical computation of the return power is then performed using the derivation of the coherent sum of the returned electric fields from the terrain (for the HH polarization only).

In Chapter 3, two different image simulation methods are developed. Both of them use in-place imaging. The first one is an idealized high resolution single frequency approach of the simulation (for a stepped frequency radar) for which the computation of the return power and the image simulation algorithms are simpler than for the real sensor approach. This approximation technique is then compared with the more complicated but realistic sensor image simulation. We also created the program architecture for both types of simulations.

In Chapters 4 and 5 the image simulation method is developed. In Chapter 4, the emphasis is put on the issues introduced by the three dimensional aspect of this simulation and the shadowing effects. The so-called pre-processing of the input data is shown to be a necessary step. One possible pre-processing is extensively derived in this chapter. In Chapter 5, we will study high level algorithms corresponding to both simulation methods. These algorithms were produced using the so-called Processing Logic Trees. They provide a clear basis to further understanding of the

developed application programs. The coherent summation derived in Chapter 2 is used in a complex image formation method which reconstructs the radar image of the observed terrain. The final purpose is to apply this method to develop an application program.

In Chapter 6, the results given by the two application programs respectively using the idealized high resolution synthesis and the real sensor simulation algorithms are studied. We present the simulations performed with the two methods for flat and non-flat terrain areas. We comment on the differences between the synthesized images and the terrain map and interpret the results.

In Chapter 7 the methods and concepts developed in this thesis are summarized. Some possible extensions and future studies are mentioned.

Chapter 2

RADAR SCATTERING MODEL

2.1 Introduction

The effects of volume scattering, even when the imaged terrain is homogeneous and flat, are visible (Goodman, 1979). The resulting image contains a granular gray-level pattern (Figure 2.1). The problem is that this visible pattern seems to have no relationship with the macroscopic properties of the illuminated terrain. Actually, the image is the superposition of the returns of many scattered wave fronts originating from different scattering points within the terrain. Since the distances traveled by these pulses can differ by several wavelengths, their superposition produces a constructive or destructive interference. This interference phenomenon causes the random speckle pattern observed in the image. This speckle often reduces the target detection predictability. Hence, the understanding of the image speckle pattern would be very useful.

In order to simulate an image, we need to model this speckle. This model will include a characterization of the properties of the terrain and also a computation of the coherent superposition of all the return pulses from the illuminated terrain. Two theoretical methods will be used to model the characteristics of each terrain type. For the coherent summation of the return pulses, the approach we will study is based upon the Radar Cross Section (RCS) of each terrain type (trees, grass...).

The RCS of an object is the apparent "size" of the object as observed by radar (Curie, 1992). The RCS is a characteristic of the terrain alone and does not depend on the range between the terrain and the radar. However, the RCS depends on the polarization and the wavelength of the radar signal as well as on the orientation of the terrain with respect to the radar line of sight.

Hence, for each terrain type we will need to have access to the RCS corresponding to the polarization and frequency of the radar for any orientation angle. These characteristics will be computed using a two-layer anisotropic random medium model (section 2.2) for some terrain types and a small perturbation method (section 2.3) for the others.

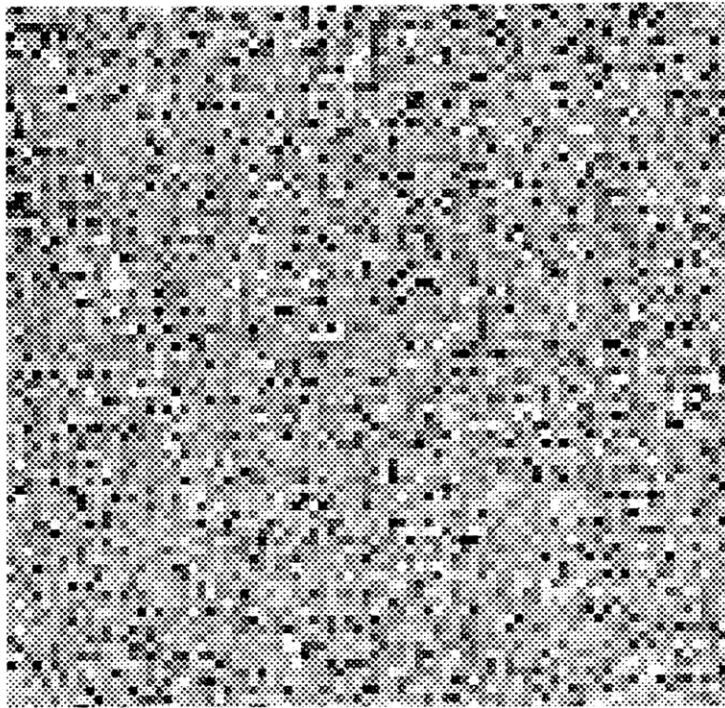


Figure 2.1: Simulated image from a flat grass field.

As it is stated in Section 1.3, we will exclusively use in-place imaging, i.e. the return power image will be derived from several observations of the terrain at different frequencies. In the following, we will assume that the radar is a stepped frequency system and that it is stationary (not in motion) relative to the observed environmental scene. The calculation of the return power from the terrain will be based on the computation the coherent sum of all the scattered wave fronts coming back from the terrain. Therefore, we will begin with the derivation of this coherent summation, for a stepped frequency system.

Even though the coherent summation of the returned wave fronts will be derived for any polarization, in this study, we will consider that the radar has only the HH polarization. Then, two different approaches will be investigated. First, we will study an idealized high resolution approach for which, even though the radar is a stepped frequency system, we will only use one frequency (reference frequency) to compute the return power from each terrain scatterer. The second study will focus on the real sensor image simulation, the return power will be computed for each scatterer at each of the radar frequencies. These data will then be processed into a high resolution image.

We will show that the first method provides an exact image for flat terrain and a good approximation for other environmental scenes. The main advantages of the idealized simulation method is that the computation it involves are much simpler than the ones the second method requires. Hence, it is very useful to study this approach first, to have a better understanding of the real radar image simulation method.

2.2 Two-Layer Anisotropic Random Medium Model

This method is a volume scattering model. The volume scattering properties of the terrain are represented by a layer of dielectric with an electrical permittivity that is a spatially random process. Beneath this layer is a non-random half-space medium (Figure 2.2).

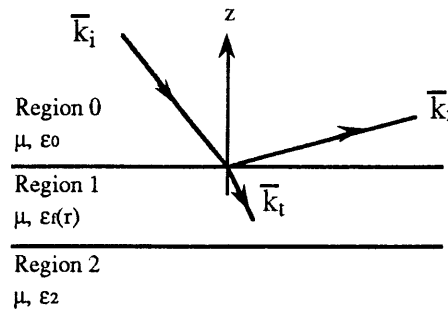


Figure 2.2: Two-layer random medium model.

The upper scattering layer is characterized by a spatially-dependent permittivity which is the sum of a mean part and a randomly fluctuating part:

$$\epsilon_1(\bar{r}) = \langle \epsilon_1 \rangle + \epsilon_f(\bar{r}) \quad (2.1)$$

We assume that $\epsilon_f(\bar{r})$ is a real, normal random variable with zero mean, which satisfies:

$$\frac{|\epsilon_f(\bar{r})|}{\langle \epsilon_1 \rangle} \ll 1 \quad (2.2)$$

i.e. the fluctuations are small compared to the mean term, and is homogeneous (spatially stationary).

Such a volume scattering model can easily be applied to terrain types that can be considered as two physical layers. For example, the ground may be considered as the bottom layer while the upper layer may be composed of grass or trees. In this simulation, we will model the backscattering properties for trees and grass using a two-layer anisotropic random medium model.

The program for the two-layer anisotropic random medium model is readily available in E.M.S.A.R.S. (Electromagnetic Model of Scattering Applied to Remote Sensing).

2.3 Small Perturbation Method

The surface scattering properties of the terrain are derived with the small perturbation method. It is generally accepted that rough surfaces can be classified into two categories: periodic roughness and random roughness. We will only use random roughness which covers a large variety of terrain.

Random surface roughness is described by a height function $f(x,y)$ as shown in Figure 2.3. This function f is considered as a normally distributed random process (with zero mean, standard deviation σ and correlation length l) in the x - y domain. The rough surface is considered as the superposition of a quasi-periodic surface $p(x,y)$ (i.e. a periodic surface with slowly varying amplitude and phase) with a small random perturbation $r(x,y)$.

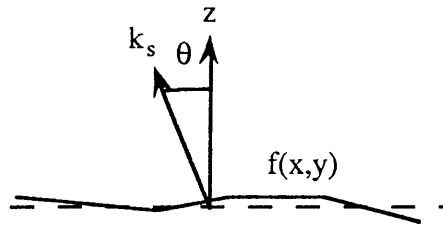


Figure 2.3: Randomly rough surface.

We have
$$f(x,y) = p(x,y) + r(x,y) \quad (2.3)$$

The small perturbation method may be applied to such a rough surface when

$$k\sigma \ll 1 \quad (2.4), \quad \text{and} \quad \sigma/l \ll 1 \quad (2.5),$$

where k is the wave number in free space, σ and l being the parameters for small roughness.

In this simulation, we will model the backscattering properties for road and water using the small perturbation method.

The program for the small perturbation method is also available in E.M.S.A.R.S.

The backscattering coefficients obtained for each terrain type using either the two-layer anisotropic random medium model or the small perturbation method will be collected into tables to which the application programs, that will be developed later in this thesis, will refer.

2.4 Computation of the Return Power

2.4.1 Coherent Summation of the Returned Pulses

This section deals with the analytical derivation of the coherent summation of the returned electric fields from a terrain for a stepped frequency system. This approach was previously investigated by Swartz (Swartz et al., 1988). The scattering coefficients referred to in this development will be obtained using either a two-layer anisotropic random medium model or a small perturbation method depending on the considered terrain type.

We consider a stepped frequency FM-CW (Frequency Modulated Continuous Wave) radar which transmits M RF pulses at frequencies ranging from f_0 to $f_0 + (M-1)\Delta f$ (i.e. M stepped frequency pulses Δf apart). Actually, each one of the pulses of the system occupies a bandwidth of approximately Δf around its theoretical frequency. Therefore, the total bandwidth B is

$$B = M\Delta f \quad (2.6)$$

The resolution in time Δt of this radar system can be related to its total bandwidth B by

$$\Delta t = \frac{1}{2B} \quad (2.7) \quad \text{i.e.} \quad \Delta t = \frac{1}{2M\Delta f} \quad (2.8)$$

We want to generate M complex electric field vectors corresponding to the backscattered radar return at the M frequencies of the system. Clutter is modeled as discrete scattering classes located in a series of N uniformly distributed range bins (Figure 2.4, page 29). We will characterize the terrain element in range bin n ($n = 0, 1, \dots, N-1$) at frequency m ($m = 0, 1, \dots, M-1$) by a unique, fully polarimetric covariance matrix computed using the scattering coefficient computation method corresponding to its terrain type. These covariance matrices will then be multiplied by Gaussian noise and the N electric field vectors are coherently combined at each of the M frequencies. The coherent high resolution range profile is then derived using an inverse Fourier transform.

We consider a plane wave incident upon the terrain medium. We can relate the scattered field \overline{E}_s to the incident field \overline{E}_i by

$$\begin{bmatrix} E_{hs} \\ E_{vs} \end{bmatrix} = \frac{e^{ikr}}{r} \begin{bmatrix} f_{hh} & f_{hv} \\ f_{vh} & f_{vv} \end{bmatrix} \begin{bmatrix} E_{hi} \\ E_{vi} \end{bmatrix} \quad (2.9)$$

where the horizontal and vertical components of the scattered field are E_{hs} and E_{vs} and those of the incident field are E_{hi} and E_{vi} .

For a reciprocal media, we have $f_{vh} = f_{hv}$. Thus, we can define the polarimetric measurement vector \overline{X} as

$$\overline{X} = \begin{bmatrix} HH \\ HV \\ VV \end{bmatrix} = \lim_{\substack{r \rightarrow \infty \\ A \rightarrow \infty}} \frac{e^{ikr}}{r} \sqrt{\frac{4\pi r^2}{A}} \begin{bmatrix} f_{hh} \\ f_{hv} \\ f_{vv} \end{bmatrix} \quad (2.10)$$

where A is the area of the illumination and r is the distance from the radar to the illuminated area.

The covariance matrix $\overline{\Sigma}$ is defined as

$$\overline{\Sigma} = \langle \overline{X} \overline{X}^\dagger \rangle = \lim_{\substack{r \rightarrow \infty \\ A \rightarrow \infty}} \frac{4\pi r^2}{A} \begin{bmatrix} \langle |f_{hh}|^2 \rangle & \langle f_{hh} f_{hv}^* \rangle & \langle f_{hh} f_{vv}^* \rangle \\ \langle f_{hv} f_{hh}^* \rangle & \langle |f_{hv}|^2 \rangle & \langle f_{hv} f_{vv}^* \rangle \\ \langle f_{vv} f_{hh}^* \rangle & \langle f_{hv} f_{hh}^* \rangle & \langle |f_{vv}|^2 \rangle \end{bmatrix} \quad (2.11)$$

where the symbols $\langle \rangle$ and \dagger denote respectively the expected value and the matrix complex transpose conjugate. The covariance matrix is normalized such that its three diagonal terms represent the radar backscattering cross section (rcs) per unit area for the HH, HV, and VV polarizations.

This covariance matrix can be obtained using the strong fluctuation theory and the distorted Born approximation (Kong, 1988), for the uniaxial two-layer random medium model.

This covariance matrix is found to be

$$\bar{\Sigma} = \sigma \begin{bmatrix} 1 & \beta\sqrt{e} & \rho\sqrt{\gamma} \\ \beta^*\sqrt{e} & e & \delta\sqrt{e\gamma} \\ \rho^*\sqrt{\gamma} & \delta^*\sqrt{e\gamma} & \gamma \end{bmatrix} \quad (2.12)$$

where

$$\sigma = \langle |HH|^2 \rangle \quad (2.13a) \quad e = \frac{\langle |HV|^2 \rangle}{\sigma} \quad (2.13b)$$

$$\gamma = \frac{\langle |VV|^2 \rangle}{\sigma} \quad (2.13c) \quad \rho = \frac{\langle HH VV^* \rangle}{\sigma\sqrt{\gamma}} \quad (2.13d)$$

$$\beta = \frac{\langle HH HV^* \rangle}{\sigma\sqrt{e}} \quad (2.13e) \quad \delta = \frac{\langle HV VV^* \rangle}{\sigma\sqrt{e\gamma}} \quad (2.13f)$$

When the media is azimuthally symmetric, covariance matrices are averages uniformly over the azimuthal angle to take into account the random orientation of the scatterers. In this case, although f_{HV} is not zero, four of the elements of the covariance matrix are null (the symmetry implies that $\beta = 0$ and $\delta = 0$ (Borgeaud, 1987)). Therefore, in this case, the covariance matrix is given by:

$$\bar{\Sigma} = \sigma \begin{bmatrix} 1 & 0 & \rho\sqrt{\gamma} \\ 0 & e & 0 \\ \rho^*\sqrt{\gamma} & 0 & \gamma \end{bmatrix} \quad (2.14)$$

In order to generate the scattered electric field vectors, we multiply the covariance matrix $\bar{\Sigma}_{mn}$ by a Gaussian white noise, for each polarimetric measurement vector \bar{X}_{mn} where

$$\bar{X}_{mn} = \begin{bmatrix} HH \\ HV \\ VV \end{bmatrix}_{mn} \quad (2.15)$$

and

$$\bar{\Sigma}_{mn} = \sigma_{mn} \begin{bmatrix} 1 & \beta\sqrt{e} & \rho\sqrt{\gamma} \\ \beta^*\sqrt{e} & e & \delta\sqrt{e\gamma} \\ \rho^*\sqrt{\gamma} & \delta^*\sqrt{e\gamma} & \gamma \end{bmatrix}_{mn} \quad (2.16)$$

and the subscripts m and n denote the frequency and range bin index, respectively.

For the simulation, we first decompose the covariance matrix for range bin n at frequency m as follows

$$\overline{\Sigma}_{mn} = \overline{\mathbf{G}}_{mn} \overline{\mathbf{G}}_{mn}^\dagger \quad (2.17)$$

where $\overline{\mathbf{G}}_{mn}$ is given by the following expression

$$\overline{\mathbf{G}}_{mn} = \sqrt{\sigma_{mn}} \begin{bmatrix} 1 & 0 & 0 \\ \beta^* \sqrt{e} & \sqrt{e(1-|\beta|^2)} & 0 \\ \rho^* \sqrt{\gamma} & \frac{\sqrt{\gamma}(\delta^* - \rho^* \beta)}{\sqrt{1-|\beta|^2}} & \sqrt{\gamma(1-|\rho|^2) - \frac{\gamma|\delta^* - \rho^* \beta|^2}{1-|\beta|^2}} \end{bmatrix}_{mn} \quad (2.18)$$

Then, the polarimetric measurement vector $\overline{\mathbf{X}}_{mn}$ can be derived as

$$\overline{\mathbf{X}}_{mn} = \overline{\mathbf{G}}_{mn} \overline{\mathbf{g}}_{mn} \quad (2.19)$$

where

$$\overline{\mathbf{g}}_{mn} = \begin{bmatrix} g_{11} + i g_{12} \\ g_{21} + i g_{22} \\ g_{31} + i g_{32} \end{bmatrix}_{mn} \quad (2.20)$$

in which g_{11} , g_{12} , g_{21} , g_{22} , g_{31} , g_{32} are independent Gaussian random variables with zero mean and a variance of 0.5.

Now, we can derive the coherent summation $\overline{\mathbf{Y}}_m$ of N scatterers for a given frequency index m. We assume that the clutter is homogeneous.

The coherent summation can be written as

$$\overline{\mathbf{Y}}_m = \sum_{n=0}^{N-1} S_n \overline{\mathbf{X}}_{mn} \exp(2i k_m r_n) \quad (2.21)$$

where \bar{Y}_m is the complex electric field at a particular frequency m . \bar{X}_{mn} is the polarimetric measurement vector corresponding to the n^{th} scatterer at the frequency m . S_n ranges from 0 to 1 depending on the amount of shadowing of the scatterer, respectively. The exponential term is here to account for the phase shift of the return due to the round trip propagation. Therefore, k_m is the wave number at frequency m ($k_m = \frac{2\pi f_m}{c}$ where f_m is $f_0 + m\Delta f$ and c is the velocity of light) and r_n is the distance from the radar to the n^{th} scatterer.

We can make the substitutions $f_m = f_0 + m\Delta f$ and $r_n = r_0 + \Delta r_n$ in which f_0 is the reference frequency of the radar, Δf is the stepped frequency increment, r_0 is the near range and Δr_n is the n^{th} scatterer phase correction term due to the round trip propagation. Thus, we find

$$\bar{Y}_m = \sum_{n=0}^{N-1} S_n \bar{X}_{mn} \exp \left[i \frac{4\pi}{c} (f_0 + m\Delta f) (r_0 + \Delta r_n) \right] \quad (2.22)$$

$$\text{i.e.} \quad \bar{Y}_m = \exp \left[i \frac{4\pi}{c} r_0 (f_0 + m\Delta f) \right] \sum_{n=0}^{N-1} S_n \bar{X}_{mn} \exp \left[i \frac{4\pi}{c} \Delta r_n (f_0 + m\Delta f) \right] \quad (2.23)$$

It is necessary to derive the expression for Δr_n to evaluate \bar{Y}_m . Let Δx be the range resolution of the system. First, a zero elevation reference plane is established. This plane is subdivided into N equally spaced points, a distance Δx apart (Figure 2.4). The environmental terrain is also subdivided into N equally spaced points, Δx apart in such a way that the ground (zero elevation) points are the vertical projections of the terrain points. These points on the terrain profile will be considered as the location of the scatterers. From each of the ground points, projections are drawn, perpendicular to the incident wave vector. These projections define the N range bins. Any two consecutive range bins are separated by a distance $\Delta x \sin \theta$ (θ is the far field incident angle). The round-trip propagation delay of the return is modified by the elevation h_n of each scatterer, in each range bin. This modification by a factor of $h_n \cos \theta$ which will introduce a distortion that can lead to the effect of overlay.

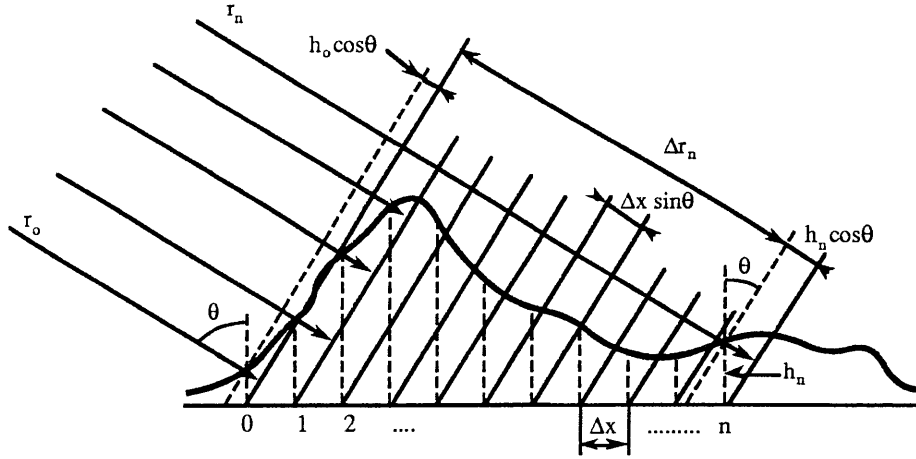


Figure 2.4: Evaluation of the phase correction term Δr_n .

The phase correction term Δr_n can be expressed as

$$\Delta r_n = n \Delta x \sin \theta - h_n \cos \theta \quad (2.24)$$

The range resolution Δx of the radar can be related to its time resolution Δt by

$$\Delta x = \frac{c \Delta t}{\sin \theta} \quad (2.25)$$

where c is the velocity of light.

We already derived that
$$\Delta t = \frac{1}{2 M \Delta f} \quad (2.26)$$

Therefore,
$$\Delta x = \frac{c}{2 M \Delta f \sin \theta} \quad (2.27)$$

Finally, the coherent sum of the returned pulses \bar{Y}_m can be expressed as

$$\begin{aligned} \bar{Y}_m = \exp \left[i \frac{4 \pi}{c} r_0 (f_0 + m \Delta f) \right] \sum_{n=0}^{N-1} S_n \bar{X}_{mn} \exp \left(i 2 \pi n \frac{f_0}{M \Delta f} \right) \\ \cdot \exp \left[-i \frac{4 \pi}{c} (f_0 + m \Delta f) h_n \cos \theta \right] \exp \left(i \frac{2 \pi}{M} mn \right) \end{aligned} \quad (2.28)$$

In the case where the radar has only one frequency, there is only one \bar{Y} , corresponding to the reference frequency. Therefore, we have $\bar{Y} = \bar{Y}_0$ which is expressed as

$$\bar{Y}_0 = \exp \left[i \frac{4 \pi}{c} r_0 f_0 \right] \sum_{n=0}^{N-1} S_n \bar{X}_{0n} \exp \left(i \frac{4 \pi}{c} f_0 \Delta x \sin \theta \right) \cdot \exp \left[-i \frac{4 \pi}{c} f_0 h_n \cos \theta \right] \quad (2.29)$$

2.4.2 Computation of the Return Power

In this study, we are only concerned about the horizontal component of the fields, i.e. the HH polarization. All the derivations present in this section are based on the results obtained in Section 2.4.1.

For a plane wave E_{hi} incident upon a terrain, the horizontal component E_{hs} of the scattered field is related to the horizontal component of the incident field by:

$$E_{hs} = \frac{e^{ikr}}{r} f_{hh} E_{hi} \quad (2.30)$$

The horizontal component X_h of the polarimetric measurement vector is defined as follows:

$$X_h = \frac{e^{ikr}}{r} \sqrt{\frac{4\pi r^2}{A}} f_{hh} \quad (2.31)$$

Substituting X_h into the first equation gives:

$$E_{hs} = X_h E_{hi} \sqrt{\frac{A}{4\pi r^2}} \quad (2.32)$$

where A is the area of the scatterer, and r is the distance between the radar and the scatterer.

Since we only consider the HH polarization case, the matrix $\overline{\overline{G}}$ defined in Section 2.4.1 reduces to

$$\overline{\overline{G}} = \sqrt{\sigma} \quad (2.33)$$

Therefore, X_h can be expressed as

$$X_h = \sqrt{\sigma} \cdot \overline{g} \quad (2.34)$$

Where σ is the horizontally polarized backscattering coefficient, and \overline{g} is a complex Gaussian random variable with zero mean and standard deviation of 0.5 (\overline{g} can be expressed as $\overline{g} = (g_0 + ig_1)/\sqrt{2}$ where g_0 and g_1 are two real Gaussian random variables with zero means and standard deviations of 1).

Using the result of the previous section, the coherent sum of scattered electric fields over the N cells within the beam width at frequency m is

$$\left[\sum_{n=0}^{N-1} E_{hs} \right]_m = \sum_{n=0}^{N-1} S_n \sqrt{\sigma_{mn}} \left[\frac{(g_0 + ig_1)}{\sqrt{2}} \right]_m E_{hi} \sqrt{\frac{A_n}{4\pi r_n^2}} e^{2i k_m r_n} \quad (2.35)$$

S_n is proportional to the amount of shadowing for the cell n. The subscript m indicates that the quantity is evaluated at frequency m. The factor $e^{2i k_m r_n}$ is the phase variation due to the round-trip propagation time between the radar and the nth scattering cell. A_n is the area of the nth scattering cell.

We can expand this equation using the notations introduced in Section 2.4.1. The result is

$$\begin{aligned} \left[\sum_{n=0}^{N-1} E_{hs} \right]_m &= \exp\left[i \frac{4\pi}{c} r_0 (f_0 + m\Delta f) \right] \sum_{n=0}^{N-1} S_n \sqrt{\sigma_{mn}} \left[\frac{(g_0 + ig_1)}{\sqrt{2}} \right]_m E_{hi} \sqrt{\frac{A_n}{4\pi r_n^2}} \\ &\cdot \exp\left(i 2\pi n \frac{f_0}{M\Delta f} \right) \exp\left[-i \frac{4\pi}{c} (f_0 + m\Delta f) h_n \cos\theta \right] \exp\left(i \frac{2\pi}{M} mn \right) \end{aligned} \quad (2.36)$$

where r_0 is the near range, f_0 is the reference frequency, Δf is the stepped frequency increment, M is the total number of frequencies of the system, h_n is the elevation of the nth scattering cell above the reference plane, and θ is the far field incident angle (see Figure 2.4, page 29).

If we let K_m equal the coherent summation as follows

$$\begin{aligned} K_m &= \exp\left[i \frac{4\pi}{c} r_0 (f_0 + m\Delta f) \right] \sum_{n=0}^{N-1} S_n \sqrt{\sigma_{mn}} \left[\frac{(g_0 + ig_1)}{\sqrt{2}} \right]_m \sqrt{A_n} \\ &\cdot \exp\left(i 2\pi n \frac{f_0}{M\Delta f} \right) \exp\left[-i \frac{4\pi}{c} (f_0 + m\Delta f) h_n \cos\theta \right] \exp\left(i \frac{2\pi}{M} mn \right) \end{aligned} \quad (2.37)$$

then the normalized scattered power is

$$[P_{hs}]_m = 4\pi r_0^2 \frac{\left| \sum_{n=0}^{N-1} E_{hs} \right|_m^2}{|E_{hi}|^2} \approx |K_m|^2 \quad (2.38)$$

The $4\pi r_0^2$ is included in the normalization to remove any range dependence. In (2.38), we assume that the range width of the radar beam is small compared to r_0 . This approximation is always valid since we consider millimeter wave systems (i.e. we need to verify $r_0 \gg c / (2 \Delta f \sin\theta)$ i.e. typically $r_0 \gg$ a few meters which is always true).

This equation is the total normalized power received from one beam width, at the m^{th} frequency.

In the case where the radar has only one frequency, we can simplify the expression above. There is only one K_m which is equal to K_0 . Therefore, the normalized received power can be expressed in this case as

$$P_{hs} = 4\pi r_0^2 \frac{\left| \sum_{n=0}^{N-1} E_{hs} \right|^2}{|E_{hi}|^2} \approx |K_0|^2 \quad (2.39)$$

where, using the previous notations, K_0 is

$$K_0 = \exp\left[i \frac{4\pi}{c} r_0 f_0\right] \sum_{n=0}^{N-1} S_n \sqrt{\sigma_{0n}} \left[\frac{(g_0 + ig_1)}{\sqrt{2}} \right] \sqrt{A_n} \cdot \exp\left[i \frac{4\pi n}{c} f_0 \Delta x \sin\theta\right] \exp\left[-i \frac{4\pi}{c} f_0 h_n \cos\theta\right] \quad (2.40)$$

Chapter 3

SIMULATION TECHNIQUES

3.1 Introduction

Without any post-processing on the returned signal, the range resolution of a single frequency radar is always limited by its beam width. The purpose of stepped frequency systems is to enhance the range resolution. Here, we are going to consider a stepped frequency FM-CW (Frequency Modulated Continuous Wave) radar which transmits M RF pulses at frequencies ranging from f_0 to $f_0 + (M-1)\Delta f$ (i.e. M stepped frequency pulses Δf apart).

The principle of stepped frequency radar imaging is to observe a terrain at M frequencies. From the M returned complex electric field vectors corresponding to the observations of the area within the radar beam at the M frequencies of the system, the associated hardware computes a high resolution image. The idea is to convert the set of M measurements of the returned electric field from the terrain within the radar beam (at M different frequencies) into a set of M returned electric fields corresponding to the returns from M equally spaced range bins within the illuminated area. Therefore, even though the range width of the radar beam is large, the effective range resolution is equal to the range width of the radar beam divided by the number of frequencies of the system, which may be very small.

In this chapter, we will analyze two different software image synthesis algorithms. First, we will study the idealized image synthesis method. As an approximation of the real sensor imaging, the synthetic range resolution of the system is the true range width divided by a factor of M and the observations are made at the reference frequency only. In this case, since the observations are only performed for one frequency, the range resolution of the idealized radar is equal to its beam range width, i.e. to the effective range resolution of the actual radar. The point is that since we directly use a high resolution beam in the computation, this approach will totally skip the issue of image distortion due to the non constant terrain elevation which is not negligible in the real sensor case. This technique does not correspond to a real image reconstruction method that can be used on the actual system. However, it yields a simplification of the equations and provides an easier understanding of the concepts. Furthermore, it will be shown in this section that this method gives a good approximation of the image and the exact solution image for the flat terrain case. The second method we will investigate corresponds to the real sensor image simulation. This image reconstruction approach will directly use the observations of the terrain at several frequencies. These samples will be processed to get the high resolution data. The image synthesis will be based on these data. A processing technique will be described in this chapter.

Eventually, we will study the architecture of an application program which will simulate radar images based on the concepts previously investigated. This phase will lead to the development of two programs which will be detailed in Chapters 4 and 5.

3.2 Simulation Techniques

In this study, will always assume that the radar system is not in motion relative to the terrain and that it has only the HH polarization.

3.2.1 Idealized Image Synthesis

As stated in the introduction, for this idealized image synthesis we assume that the range width $w_{r \text{ ideal}}$ of the idealized radar beam is equal to the effective range resolution Δx of the actual system. The observations will be made at the reference frequency only. Therefore, in this section, even though the radar is a stepped frequency system, we will use single frequency results. Assume that the range width of the actual radar beam is given by

$$w_r = \frac{c}{2 \Delta f \sin \theta} \quad (3.1)$$

where c is the velocity of light, Δf is the stepped frequency increment and θ is the far field incident angle (Figure 2.4, page 29), the effective range resolution of the actual radar is given in (2.27) and we have

$$\Delta x = \frac{w_r}{M} = w_{r \text{ ideal}} \quad (3.2)$$

where Δx is the range resolution and M is the total number of frequencies of the system.

Thus, we will consider the range width of the idealized radar beam to be equal to that of the actual radar divided by the number of frequencies of the real system (Figure 3.1).

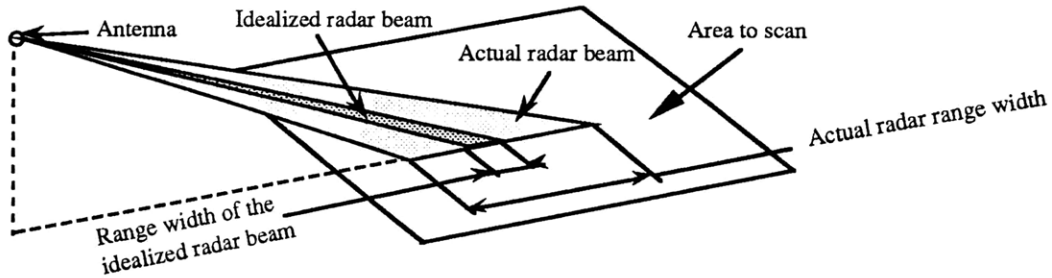
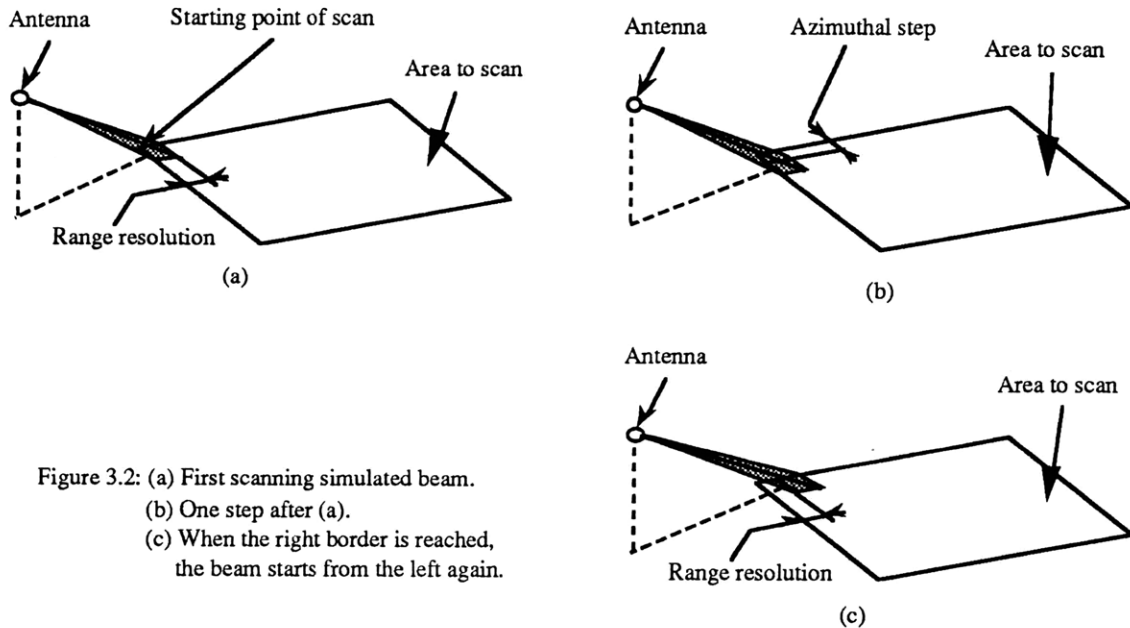


Figure 3.1: Range width of the idealized radar beam.

Using this definition, since the observation will only be performed at the reference frequency, the idealized range resolution Δx_{ideal} will be equal to the idealized beam range width and therefore equal to the actual radar effective range resolution Δx .

Once this important parameter of the radar has been defined, the following simulation method may be applied to this idealized radar:

- The terrain is scanned by the radar beam. This beam has the same azimuth width as the actual radar beam, its range width is the one defined above. The starting point of scan is the near left corner of the terrain. The scan goes from left to right at constant range with a chosen azimuthal step. One the right border to the terrain reached, the beam starts from the left again, one range resolution further (Figure 3.2).
- For each step, we compute the return power from all the scatterers within the beam.
- The return image is the image in shades of gray of the whole array of return powers obtained during the scan of the terrain.



For the computation of the return power, we use the derivation performed in Section 2.4.2.

The return power from the scatterers within the beam is given in (2.39) and (2.40).

Substituting in (2.40) the expression for Δx found in (2.27) we can express the return power as

$$P_{hs} = 4\pi r_0^2 \frac{\left| \sum_{n=0}^{N-1} E_{hs} \right|^2}{|E_{hi}|^2} \approx |K_0|^2 \quad (3.3)$$

with

$$K_0 = \exp\left[i \frac{4\pi}{c} r_0 f_0\right] \sum_{n=0}^{N-1} S_n \sqrt{\sigma_{0n}} \left[\frac{(g_0 + ig_1)}{\sqrt{2}} \right] \sqrt{A_n} \\ \cdot \exp\left(i \frac{4\pi n}{c} f_0 \Delta x \sin\theta\right) \exp\left[-i \frac{4\pi}{c} f_0 h_n \cos\theta\right] \quad (3.4)$$

where r_0 is the distance from the radar to the illuminated region, f_0 is the reference frequency, S_n ranges from 0 to 1 depending on the amount of shadowing of the scatterer, σ_{0n} is the horizontally polarized backscattering radar cross section of the n^{th} scatterer at frequency f_0 , g_0 and g_1 are two real Gaussian random variables with zero means and standard deviations of 1. A_n is the area of the n^{th} scatterer, Δx is the range resolution of the idealized radar, h_n is the elevation of the n^{th} scatterer above the reference plane and θ is the far field incident angle.

The return power image can be computed from the whole array of observed return powers.

3.2.2 Real Sensor Image Simulation

In the real radar system, high-range-resolution images are not obtained using the previous approach. The issue is that the actual sensor can only receive the returned electromagnetic fields from the illuminated terrain within its beam. For an observation at one frequency, the range resolution is equal to the range width of the radar beam. It is only by using several measurements of the same region at several frequencies that one can improve the range resolution and obtain high resolution images. Therefore, it is obvious that the idealized method does not correspond to a real processing possibility. The actual radar image synthesis must only use what the real radar system can measure.

Therefore, we consider a stepped frequency radar which transmits M RF pulses at frequencies ranging from f_0 to $f_0 + (M-1)\Delta f$. The range width w_r of the radar beam is given by equation (3.1) and according to (2.27) we have

$$w_r = \frac{c}{2 \Delta f \sin\theta} = M\Delta x \quad (3.5)$$

where c is the velocity of light, Δf is the stepped frequency increment and θ is the far field incident angle (Figure 2.4, page 29).

The simulation method is nearly the same as for the idealized case. The only differences are that now the beam range width is M times as large as it is for the ideal method and that for each step we observe the terrain at M different frequencies instead of only f_0 . The simulation process is the following:

- The terrain is scanned by the radar beam. This beam has the same dimensions as the actual beam. The starting point of scan is the near left corner of the terrain. The scan goes from left to right at constant range with a chosen azimuthal step. Once the right border to the terrain reached, the beam starts from the left again, one range resolution further (Figure 3.2, page 36).
- At each step, we store the returned complex electric field vector which is the sum of all the scattered pulses from the terrain within the beam at each one of the M frequencies of the system.

- The array of returned electric fields, each one corresponding to one frequency, observed for one step is then processed to get the high resolution return power pattern. This processing improves the range resolution of the system using the information contained in the returned electric fields observed at M different frequencies for the same illuminated region.
- The return image is the image in shades of gray of the whole array of high resolution return powers obtained after processing the returns evaluated at M different frequencies.

For the computation of the sum of the electric fields, we use the derivation performed in Section 2.4.2. The sum of all the electric fields scattered from the terrain within the beam at the m^{th} frequency is given in (2.36) as

$$\left[\sum_{n=0}^{N-1} E_{hs} \right]_m = \exp \left[i \frac{4 \pi}{c} r_0 (f_0 + m \Delta f) \right] \sum_{n=0}^{N-1} S_n \sqrt{\sigma_{mn}} \left[\frac{(g_0 + i g_1)}{\sqrt{2}} \right]_m E_{hi} \sqrt{\frac{A_n}{4 \pi r_n^2}} \cdot \exp \left(i 2 \pi n \frac{f_0}{M \Delta f} \right) \exp \left[-i \frac{4 \pi}{c} (f_0 + m \Delta f) h_n \cos \theta \right] \exp \left(i \frac{2 \pi}{M} m n \right) \quad (3.6)$$

where r_0 is the distance from the radar to the illuminated region, f_0 is the reference frequency, S_n ranges from 0 to 1 depending on the amount of shadowing of the scatterer, σ_{mn} is the horizontally polarized backscattering radar cross section of the n^{th} scatterer at frequency $f_0 + m \Delta f$, g_0 and g_1 are two real Gaussian random variables with zero means and standard deviations of 1. A_n is the area of the n^{th} scatterer, h_n is the elevation of the n^{th} scatterer above the reference plane and θ is the far field incident angle.

For the same illuminated area, M returned complex electric fields are computed as described above at the M different frequencies of the system. Once calculated, these M electric fields are processed to get the high resolution return power pattern (Figure 3.3 and Table 3.1). One possible conversion process will now be investigated.

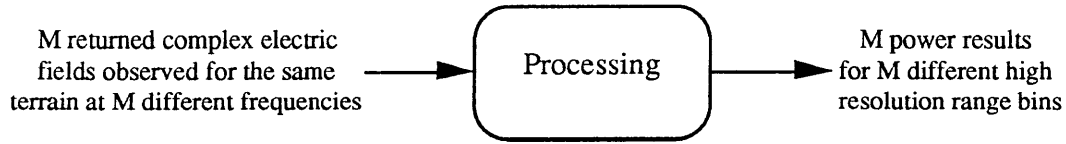
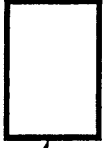
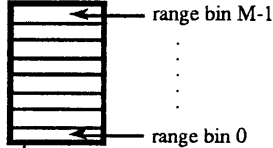


Figure 3.3: Actigram for the conversion processing.

Table 3.1: Conversion processing.

Input data	Reference cell	Output data	Reference cells
Returned \underline{E} at M frequencies f_0 to $f_0 + M\Delta f$	 <p>Area illuminated by the actual radar beam</p>	Power corresponding to each one of M range bins within the beam	 <p>Area illuminated by the actual radar beam</p>

Conversion Processing

In this paragraph, we want to derive the return power for the M high resolution range bins from the M complex electric fields scattered from the same area at the M frequency of the radar system. Actually, we want to get a result that is as close to the one obtained in the idealized case as possible.

Let \underline{E}_m be the scattered electric field (horizontal component only, since we consider only the HH polarization) evaluated at the m^{th} frequency. Let n be the index of the range bins. Actually, \underline{E}_m is the coherent sum of M scattered electric fields \underline{E}_{0n} at frequency f_0 (the return difference between two frequency steps is only due to the propagation time difference) from each of the M range bins. \underline{E}_{0n} are the measured electric fields for the idealized method.

We have

$$\underline{E}_m = \sum_{n=0}^{M-1} S_n \underline{E}_{0n} \exp(2 i k_m r_n) \quad (3.7)$$

where S_n ranges from 0 to 1 depending on the amount of shadowing of the scatterer, the exponential term is there to take into account the phase variation due to the round-trip propagation delay between the radar and the n^{th} scatterer, k_m is the wave number at frequency m and r_n is the distance from the radar to the n^{th} scatterer.

We have

$$k_m = \frac{2 \pi f_m}{c} \quad (3.8)$$

where f_m is $f_0 + m\Delta f$ and c is the velocity of light.

So, if we let

$$k_0 = \frac{2 \pi f_0}{c} \quad (3.9)$$

We get

$$k_m = k_0 + m \Delta k \quad (3.10)$$

where Δk is a constant with

$$\Delta k = \frac{2 \pi \Delta f}{c} \quad (3.11)$$

In Section 2.4.1, we have shown that

$$r_n = r_0 + \Delta r_n = r_0 + n \delta r_n \quad (3.12)$$

with

$$\delta r_n = \Delta x \sin\theta - \frac{h_n \cos\theta}{n} \quad (3.13)$$

where r_0 is the near distance from the radar to the illuminated region, Δf is the range resolution of the system, h_n is the elevation of the n^{th} scatterer above the reference plane and θ is the far field incident angle.

An important issue is that δr_n is not a constant (except for flat terrain). Therefore it will introduce a distortion which could lead to overlay (Figure 3.4). The actual sensor will not be able to detect this distortion.

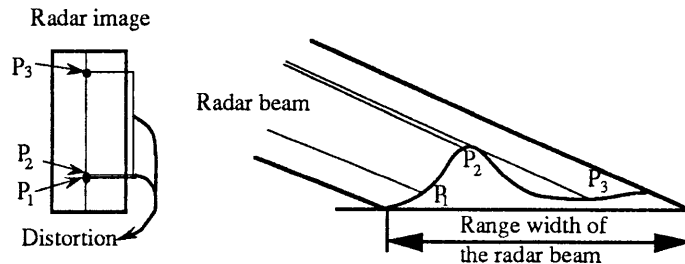


Figure 3.4: Image distortion.

Therefore, we have

$$\underline{E}_m = \sum_{n=0}^{M-1} S_n \underline{E}_{0n} \exp\left(2 i (k_0 + m \Delta k) (r_0 + n \delta r_n)\right) \quad (3.14)$$

when we factor out the term $\exp\left(2 i (k_0 + m \Delta k) r_0\right)$, we get the following expression

$$\underline{E}_m = \exp\left(2 i (k_0 + m \Delta k) r_0\right) \sum_{n=0}^{M-1} S_n \underline{E}_{0n} \exp\left(2 i (k_0 + m \Delta k) n \delta r_n\right) \quad (3.15)$$

Let

$$\underline{T}_m = \underline{E}_m \exp\left(-2 i (k_0 + m \Delta k) r_0\right) \quad (3.16)$$

and

$$\underline{T}_{0n} = S_n \underline{E}_{0n} \exp\left(2 i k_0 n \delta r_n\right). \quad (3.17)$$

With these notations, we obtain

$$\underline{T}_m = \sum_{n=0}^{M-1} \underline{T}_{0n} \exp\left(2 i m \Delta k n \delta r_n\right) \quad (3.18)$$

This expression looks like a Fourier transform. However, since δr_n is not a constant, this equation is only an approximate Fourier transform. The only case for which we can derive an exact expression is for a flat terrain (δr_n is a constant). We cannot, in general, exactly recover the \underline{T}_{0n} from the measurements of the \underline{T}_m (this equation cannot be inverted by the actual radar computer since the δr_n are unknown).

It is necessary that we make an approximation to derive the results. We will consider all throughout the derivation that the factor δr_n is a constant for all n . The radar image will be distorted. The actual radar imaging system will show the same distortion. Hence, we will assume in the following that δr_n is the constant δr , with

$$\delta r = \Delta x \sin\theta \quad (3.19)$$

Then the expression for \underline{T}_m becomes

$$\underline{T}_m = \sum_{n=0}^{M-1} \underline{T}_{0n} \exp\left(2 i m \Delta k n \delta r\right) \quad (3.20)$$

This expression is a Fourier transform if and only if we have the following relation

$$2 M \Delta k \delta r = 2 \pi. \quad (3.21)$$

Since we have

$$\Delta k = \frac{2 \pi \Delta f}{c} \quad (3.22) \quad \text{and} \quad \delta r = \Delta x \sin \theta \quad \text{with} \quad \Delta x = \frac{c}{2 M \Delta f \sin \theta}, \quad (3.23)$$

equation (3.21) is satisfied and therefore it is an exact Fourier transform. Therefore, when δr is a constant (flat terrain), the real sensor simulation gives the same result as the idealized approach. A better way to think about it is to claim that for the flat terrain case the idealized method gives the exact result.

We can express \underline{T}_{0n} as a function of \underline{T}_m as

$$\underline{T}_{0n} = \frac{1}{M} \sum_{m=0}^{M-1} \underline{T}_m \exp\left(-\frac{2 i \pi m n}{M}\right) \quad (3.24)$$

which is equivalent to

$$S_n \underline{E}_{0n} \exp\left(2 i k_0 n \delta r_n\right) = \frac{1}{M} \sum_{m=0}^{M-1} \underline{E}_m \exp\left(-2 i (k_0 + m \Delta k) r_0\right) \exp\left(\frac{2 i \pi m n}{M}\right) \quad (3.25)$$

The normalized return power P_n from the n^{th} range bin is given by

$$P_n = 4\pi r_0^2 \frac{\left| S_n \underline{E}_{0n} \exp\left(2 i k_0 (r_0 + n \delta r)\right) \right|^2}{|E_{hi}|^2} \quad (3.26)$$

where $\exp\left(2 i k_0 (r_0 + n \delta r)\right)$ accounts for the delay due to the round-trip propagation between the radar and the range bin. $|E_{hi}|^2$ is the magnitude squared of the incident wave electric field.

Since the magnitude of $\exp\left(2 i k_0 (r_0 + n \delta r)\right)$ is unity, we also have

$$P_n = 4\pi r_0^2 \frac{|S_n \underline{E}_{0n}|^2}{|E_{hi}|^2} \quad (3.27)$$

i.e.

$$P_n = 4\pi r_0^2 \frac{\left| \frac{1}{M} \sum_{m=0}^{M-1} E_m \exp(-2i(k_0 + m\Delta k)r_0) \exp\left(\frac{2i\pi m n}{M}\right) \right|^2}{|E_{hi}|^2} \quad (3.28)$$

Conclusion

This derivation shows that the idealized image synthesis method provides the exact solution in the flat terrain case. For the real sensor simulation, there is no other solution than applying the above relation even when δr_n is not a constant. In this case, the relationship between \underline{T}_{0n} and \underline{T}_m is an approximate Fourier transform. The general result for the real sensor image synthesis is

$$P_n = 4\pi r_0^2 \frac{\left| \frac{1}{M} \sum_{m=0}^{M-1} E_m \exp(-2i(k_0 + m\Delta k)r_0) \exp\left(\frac{2i\pi m n}{M}\right) \right|^2}{|E_{hi}|^2} \quad (3.29)$$

3.3 Simulation Program Architecture

In the previous sections we investigated two radar image synthesis methods. The final purpose of this study is to implement these methods on a computer. Therefore, this section will deal with the analysis of an application program based on the simulation techniques previously described. The radar image simulation function will be divided into several linked function blocks. We will examine the links between these function blocks rather than their detailed content. Some aspects of the simulation such as the scanning procedure were previously examined. A specific study of the main blocks will be performed later in this thesis.

The radar image synthesis system can be sketched as a block that receives a map of the terrain and several parameters of the radar as input as produces the synthesized image (Figure 3.5).

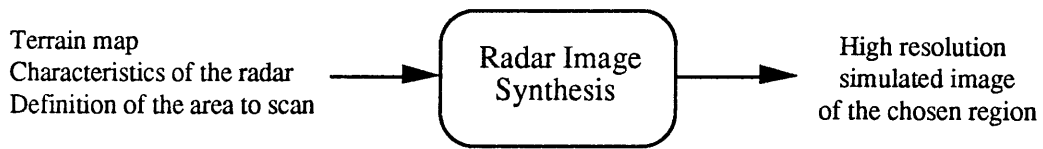


Figure 3.5: Actigram for the radar image synthesis system.

For the derivation of the return power from a terrain (cf. Sections 3.2.1 and 3.2.2), we need several data other than the ones stated in Figure 3.5. Actually, the computation of the return power includes a complex Gaussian weighting, and the use of the radar cross section of the scatterers. Therefore, this "Radar Image Synthesis" function must contain a generation of Gaussian random variables for the weighting, and also a generation of look-up tables for the radar cross sections. These look-up tables will be generated from the data collection performed using E.M.S.A.R.S. modules mentioned in Sections 2.2 and 2.3.

Hence, the "Radar Image Synthesis" block can be split into several function blocks as shown in Figure 3.6. An actigram is always read from left to right. The arrows entering boxes from their left side represent data input. The arrows entering boxes from the top indicate that this function is activated by the end of the previous one. The arrows entering a box by its bottom indicates resources used by the box. Hence, for example, the box "Generation of look-up tables of

radar cross sections versus elevation angle" is activated by the arrival of the input data, and does not directly use the input data but the radar cross section data collection instead.

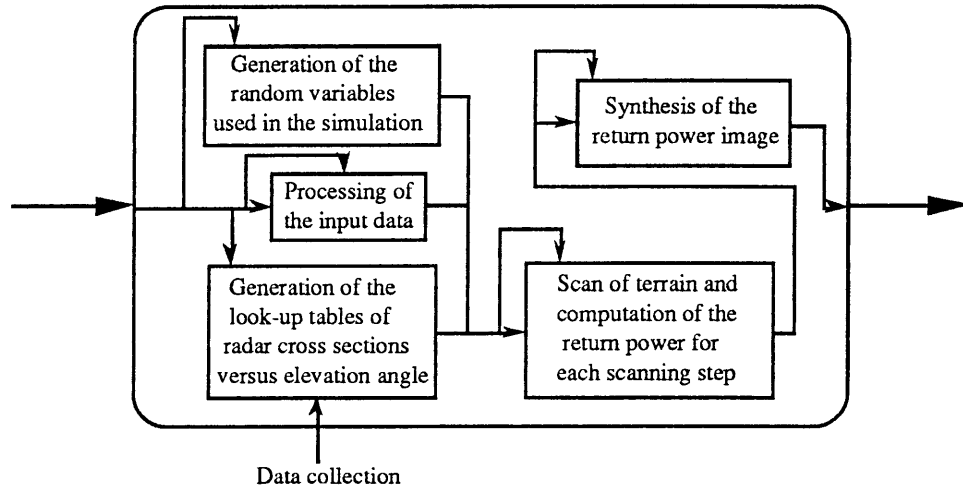


Figure 3.6: Actigram for radar image synthesis.

The actigram in Figure 3.6 is sufficiently low-level to start the development of an application program on this basis. It would be very difficult to split the function blocks it contains without losing the general nature required for an architecture diagram.

The functions corresponding to the "Processing of the input data" and the "Scan of the terrain and computation of the return power at each step" will be developed in Chapter 4 and 5, respectively.

Chapter 4

PRE-PROCESSING OF THE INPUT DATA

4.1 Introduction

In this chapter, we will focus on the data processing aspect of the simulation. We first need to specify the input data. Our main input is a digitized aerial map of the terrain surrounding the area to be imaged and the elevation data corresponding to the terrain map. For this input map, we need to have a table containing the relationship between the input map gray level and the corresponding terrain type (trees, grass, road...). The radar has to be described by its reference frequency, number of steps, the stepped frequency increment, and the beam azimuth width. Obviously, the scale of the terrain map, the dimensions of the area to image, its location and the location of the radar sensor have to be known.

The coherent summation of the electric fields coming back from a scattering terrain require the knowledge of the backscattering cross section of any part of the terrain at any elevation angle. We need a table giving the radar cross section versus the elevation angle of the radar for each considered terrain type. As it is stated in Section 3.3, these data will be collected a priori, using the E.M.S.A.R.S. programs. These tables will be used as a database.

The objective of the computer program is to generate several data files from these input data. First, the image of the chosen area will be synthesized. The power returns will be distributed automatically among 256 gray levels to create an image and one of the output files will contain the histogram of this distribution. The program will also provide a complete list of all the return powers from the scanned region.

The aerial image of the terrain is a computer image and therefore it is discrete (so is the topography). Hence the terrain image is an array of terrain cells and not a continuous terrain. The issue is that, since the terrain is not flat, the exact shape of the terrain is not completely determined by the input files (they only provide samples of the shape of the terrain). The range resolution of the radar is usually smaller than the resolution of the input terrain map. Thus, it is necessary to make an interpolation between the points given in the input files to compute the returned power within the surface of a radar beam.

For this purpose, instead of directly using the terrain cells given in the input files, we will partition the terrain map into another set of cells. Theoretically, the range dimension of these cells could be, in the upper limit, equal to the range width of the radar beam. However, another important issue (which will be mentioned later in this document) due to the three dimensional aspect of the terrain reveals that it is necessary to use smaller cells to obtain acceptable results. Hence, for the range dimension of the cells, we will use the range width of the radar beam divided by an oversampling factor. In the following, these cells will be referred to as small cells.

4.2 Pre-Processing Method

The terrain map is scanned to determine the characteristics of each small cell which will be used to compute the return power. This first scanning process will use the following method :

- The total azimuthal width of the area to be scanned is partitioned into radial lines belonging to the lowest horizontal plane (null elevation). The angle between two contiguous lines is equal to the azimuthal step (Figure 4.1).

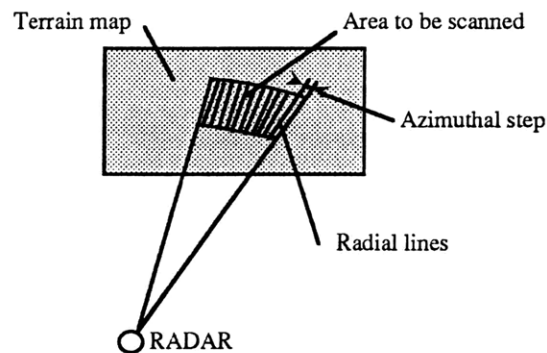


Figure 4.1: The total azimuthal width of the area to be scanned is divided into radial lines.

- Each of the preceding lines is discretized into a set of points. The sampling step is equal to the radar range width divided by the oversampling factor. The larger this oversampling factor, the smaller the cells and the more accurate the computed returned power.

- Each one of these points will be considered as the projection on the horizontal plane of the center of a non-flat small cell (Figures 4.2 and 4.3). A procedure will be applied to each of these points in order to determine the characteristics (Table 4.1) of each non-flat cell. The purpose of the pre-processing is to create a terrain map with a better resolution than the one of the input map. There will be several small cells between two terrain cells.

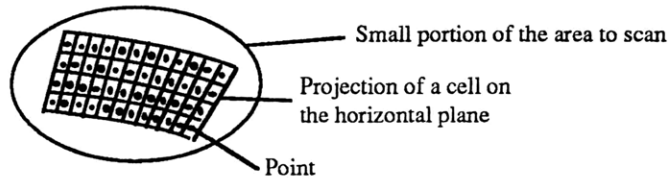


Fig 4.2: Definition of the horizontal projection of the small cells.

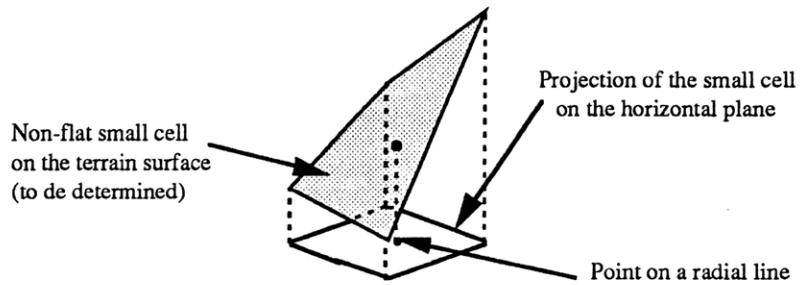


Fig 4.3: Definition of the non-flat small cells.

Table 4.1: Characteristics computed for each cell.

<i>Characteristics</i>	<i>Type</i>
Coordinates of the center point	Real x, y, z
Slope of the cell in the incident wave plan	Real in degrees
Terrain type of the cell	Integer [0..256]
Type of shadowing	Integer 0, 1, 2

The following procedure is applied to each of these points to determine the characteristics of the corresponding three dimensional cell:

* The program detects in which terrain cell the current point is (Figure 4.4).

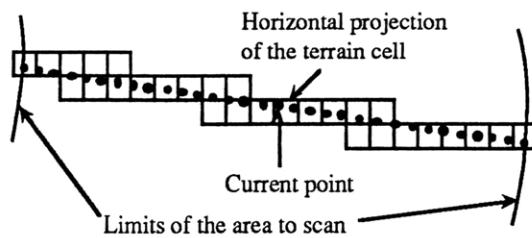


Figure 4.4: The program determines in which terrain cell the current point is.

* The global coordinates of the two points I_1 and I_2 of intersection of the vertical plane V including the radial line with the four sides of the terrain cell are computed (Figure 4.5). For further reference, I_1 is the closest to the radar sensor of the two of them.

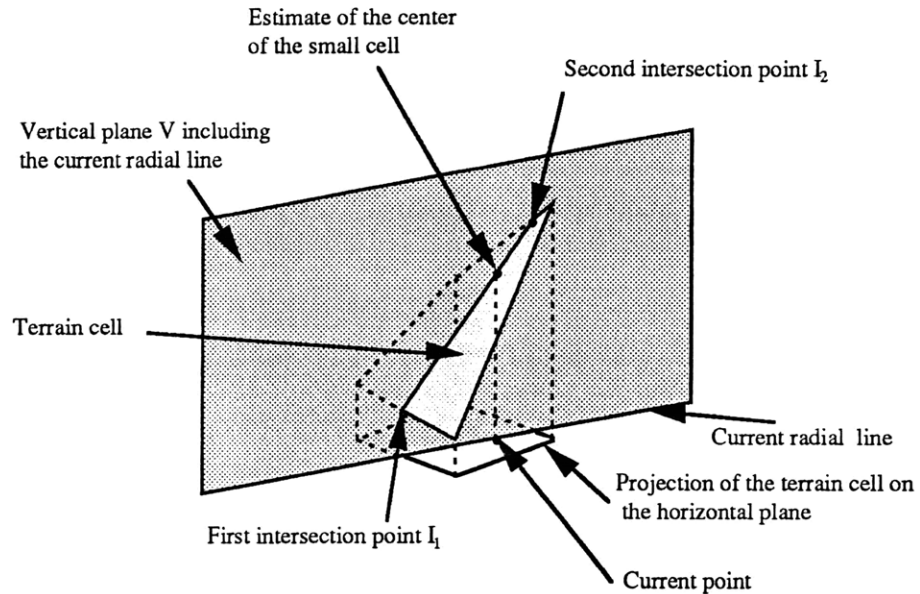


Figure 4.5: Definition of the two points of intersection I_1 and I_2 of the vertical plane V and the four sides of the terrain cell.

* The center of the small cell is assumed to be the vertical projection of the current point on the terrain cell. This point belongs to the line of the two previously computed intersection points I_1 and I_2 (Figure 4.5).

* The slope of the line including the two points I_1 and I_2 is the slope of the small cell in the vertical plane V. This slope will be used in the computation of the returned power.

* The terrain type (trees, grass,...) of the small cell is chosen to be the terrain type of the closest terrain point given in the input file.

* The shadowing parameter of the cell is computed. This parameter is used to distinguish between shadowed and not shadowed points and also between two types of shadowing. It is defined to take the values (Figure 4.6 (a) and (b)):

- 0: not in a shadow.
- 1: in a shadow, back of a hill.
- 2: terrain in a shadow, facing the beam.

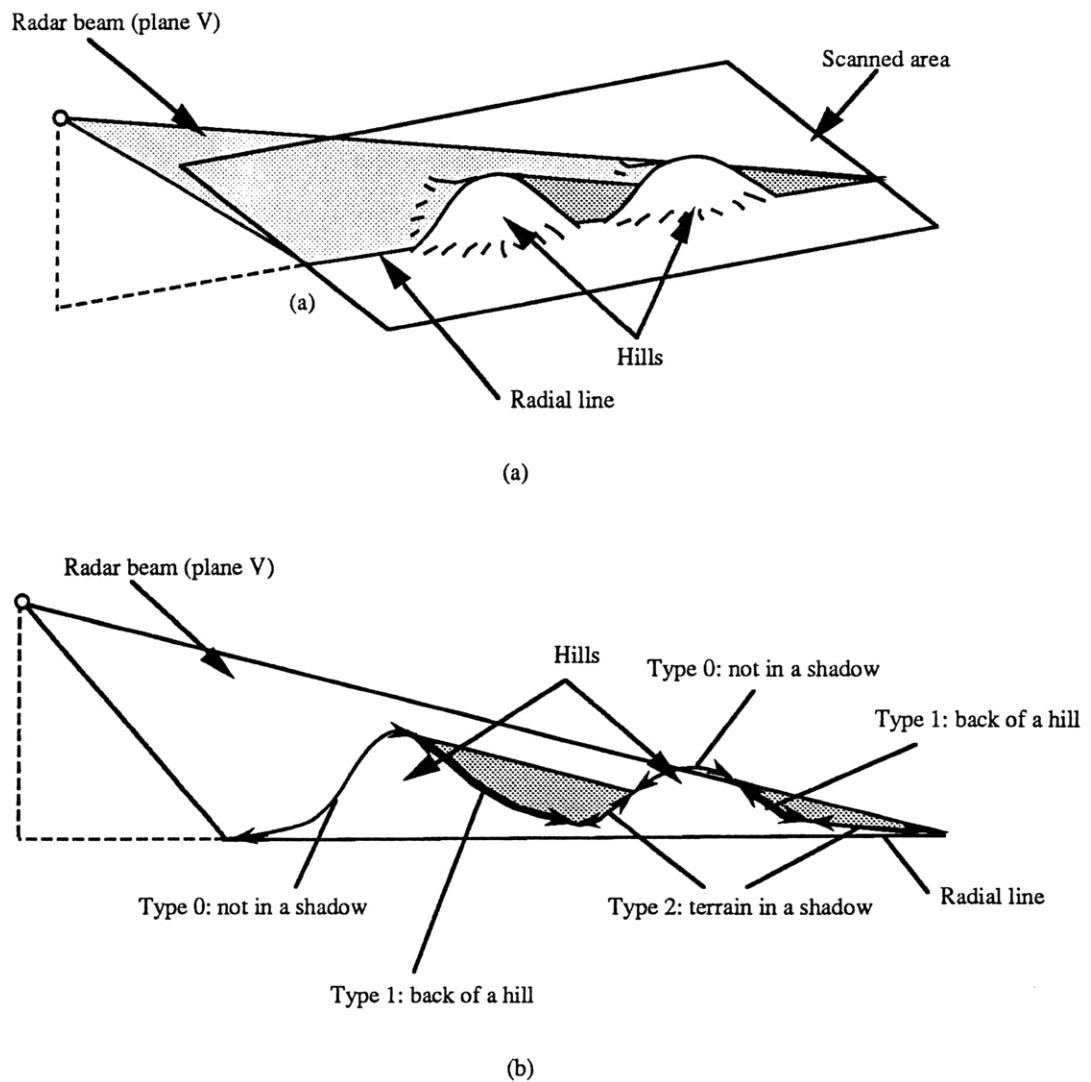


Figure 4.6 (a) and (b): The shadowing parameter of the cell can take three values.

4.3 Mathematical Formulation

4.3.1 Definitions

First, let us define the coordinate system, which is shown in Table 4.2 and Figure 4.7.

Table 4.2: Definitions.

<i>Parameter</i>	<i>Notation</i>
Global coordinates of the radar sensor	(x_S, y_S, z_S)
Global coordinates of the terrain map origin	$(x_T, y_T, 0)$
Global coordinates of the current point	(x_P, y_P, z_P)
Angle between the current radial line and the north	beam_angle
Distance between the radar and the current point	dist

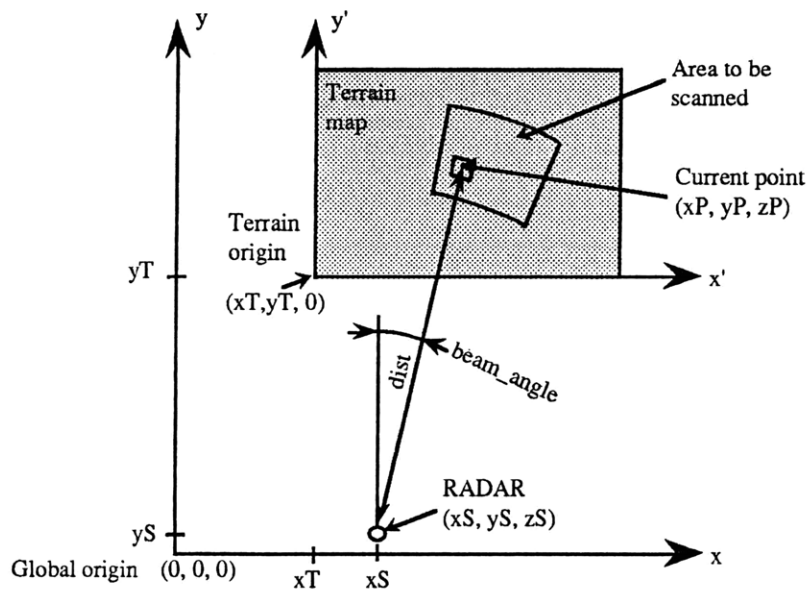


Figure 4.7: Geometrical definitions.

4.3.2 Computation of the characteristics of the small cells

a) Coordinates of I_1 and I_2

The two points I_1 and I_2 are the points of intersection of the vertical plane V and the four sides of the terrain cell (Figure 4.5, page 51). In order to compute the global coordinates of these two points, we first need to calculate the coordinates of the points of intersection between the vertical plane V and the four lines including the sides of the terrain cell (Figure 4.8). In general there are four of these points the only exception being when the vertical plane V is parallel to two of the sides of the terrain cell (in that case there are only two intersection points which are I_1 and I_2 , the following algorithm is useless). The analytical calculation of the coordinates of the intersection point between a vertical plane and a line is derived in Appendix A. We apply this method four times to the vertical plane V and each one of the four lines including a side of the terrain cell. Thus, we get the coordinates of the four desired intersection points.

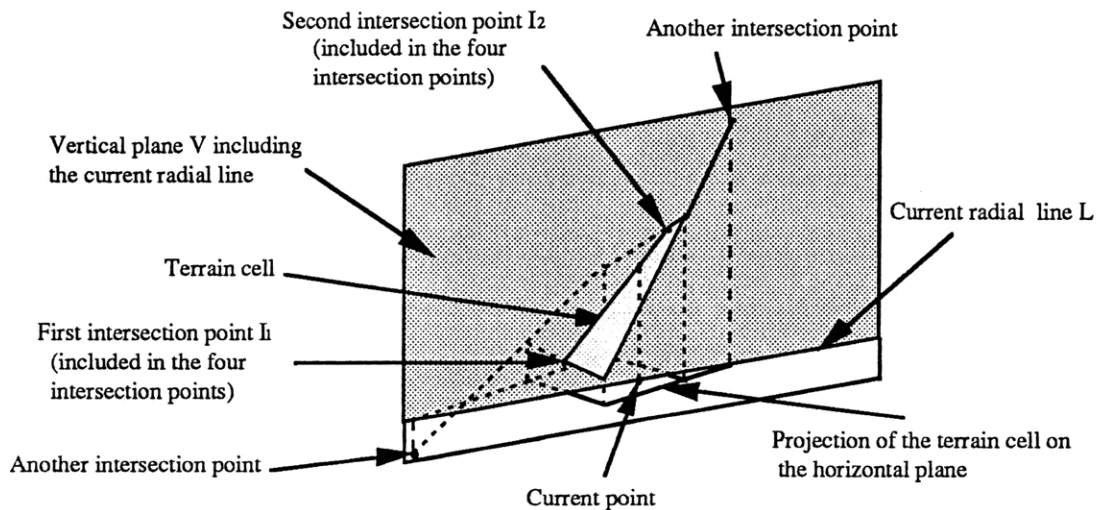


Figure 4.8: Definition of the four points of intersection between the vertical plane V and the four lines including the sides of the terrain cell.

Only two of these points are actually I_1 and I_2 , we need to determine which ones of the four points they are. The discrimination algorithm is obvious: I_1 and I_2 are the only two points belonging to the line segments of the sides of the terrain cell (Figure 4.9).

Once this discrimination done, we know the global coordinates of the two points I_1 and I_2 . The point I_1 is assumed to be the closest to the radar sensor of the two of them.

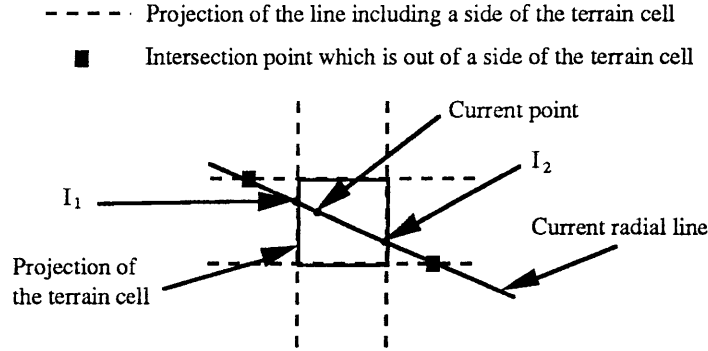


Figure 4.9: Projection on the horizontal plane. Discrimination criterion used to find I_1 and I_2 .

b) Coordinates of the center of the small cell

We already know by definition the x and y components (x_P and y_P) of the center of the small cell. They are defined by:

$$x_P = x_S + \text{dist} \sin(\text{beam_angle}) \quad (4.1)$$

and
$$y_P = y_S + \text{dist} \cos(\text{beam_angle}) \quad (4.2)$$

The z component z_P of this point is easily obtained since it belongs to the line segment defined by I_1 and I_2 .

Assuming $I_1 = (x_1, y_1, z_1)$ and $I_2 = (x_2, y_2, z_2)$, we have:

$$z_P = z_1 + (z_2 - z_1) \frac{\sqrt{(x_P - x_1)^2 + (y_P - y_1)^2}}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \quad (4.3)$$

c) Slope of the small cell in the vertical plane V

The slope of the small cell in the vertical plane V is defined as the slope of the line segment defined by I_1 and I_2 in V. The slope is considered negative if I_1 is higher than I_2 .

Assuming $I_1 = (x_1, y_1, z_1)$ and $I_2 = (x_2, y_2, z_2)$, we have:

$$\text{slope} = \text{Arctan} \left[\frac{(z_2 - z_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \right] \quad (4.4)$$

d) Terrain type

The terrain type of the cell is chosen to be the terrain type of the closest corner of the current terrain cell (Figure 4.10).

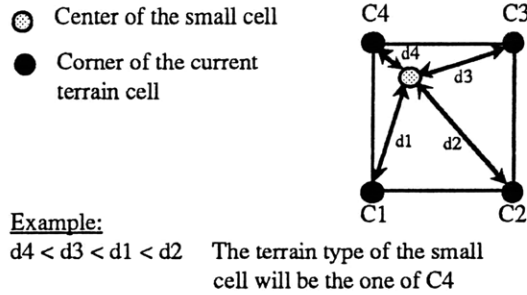


Figure 4.10: Definition of the terrain type of the small cells.

e) Shadowing parameter

This parameter is determined according to the relative heights of the current point and all the preceding points on the same radial line. To be consistent, the shadowing parameter is computed for the elevations of all the points of a radial line from the beginning of the terrain map. This extra computation is necessary unless we assume that the area to be scanned is not, even partly, in the shadow of the terrain area between it and the radar.

We assume that the shadowing type of a cell is the one of its center point. Thus, a small cell is considered to be in a shadow or not, but it is never considered as partially in a shadow.

The shadowing parameter is computed as follows:

- For each small cell, we compute the minimum height for the center point of the next cell not to be in a shadow. It is directly computed from the elevation of the current cell if this one is not in a shadow. Otherwise, it is computed from the current minimum height.
- If the elevation of the center of the current cell is less than the current minimum height then the current cell is assumed to be totally in a shadow. Hence its shadowing parameter is no longer equal to 0 but can be 1 or 2.

- If the shadowing parameter of the current cell is not equal to 0 then it is set to 1 if the sum of the depression angle (angle between the radar line of sight to the cell and the horizontal plane) and the slope of the cell is negative (Figure 4.11). The shadowing parameter is otherwise set to 2.

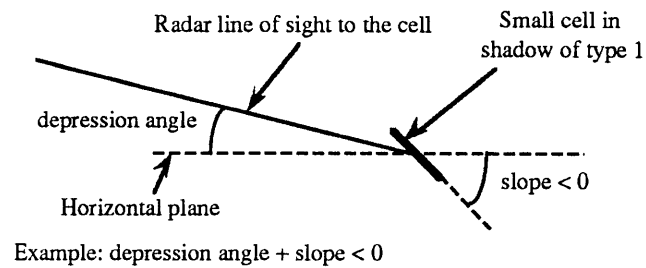


Figure 4.11: Case where the shadowing parameter is equal to one.

Once this pre-processing of the input data is done, the coherent superposition of all the returns to the radar can be performed. This computation is the point developed in Chapter 5.

Chapter 5

COMPUTATION OF THE SIMULATED RADAR IMAGE

5.1 Introduction

Chapter 3 shows the high level structure of terrain scanning and image synthesis methods for both the idealized and the real sensor simulation techniques. The pre-processing of the input data is described in Chapter 4. The purpose of this chapter is to develop an implementable method based on these concepts for each simulation technique. In order to be consistent, the similarity between the idealized and the real case approaches will be preserved as long as possible.

The studied terrain is a priori not flat, therefore several issues must be taken into account. First, some regions of the terrain may be in a shadow (this is one of the reasons why the terrain map is pre-processed). Second, since the terrain points are not on the reference plane (zero elevation), a special scanning has to be performed to Figure out which point is in which range bin. One possible scanning process will be extensively described in this chapter. We will explain the computation of the return power in both idealized and real sensor cases. We will go into as many details as possible, the very next step being the implementation on a computer of the radar image simulation process.

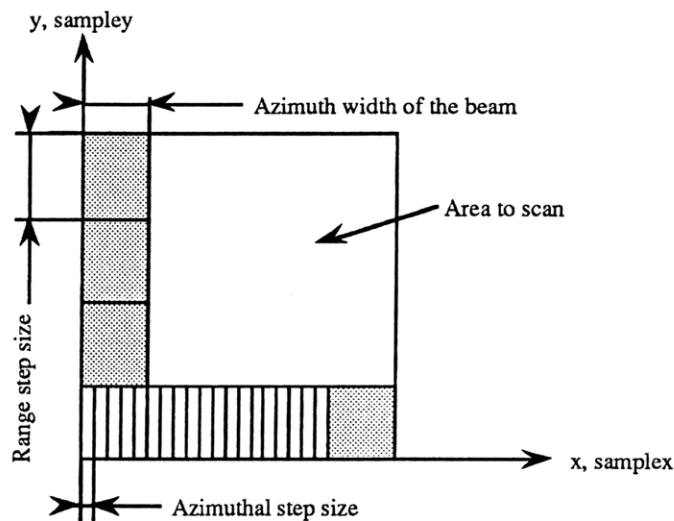
5.2 Method, Algorithms

5.2.1 Introduction, Definitions

According to the actigram given in Section 3.3, after the pre-processing of the input data, once the array of random variables and the radar cross section look-up tables are generated, the terrain is scanned to compute the radar return power and synthesize the image. The scanning method was introduced in Section 3.2. We now investigate it more deeply.

First, we need to introduce several notations to simplify the references for the reasoning. These definitions are in Table 5.1, next page (see also Figure 5.1).

The radar is far from the imaged area and the radar beam widths (range and azimuth) are small. Therefore, to simplify the reasoning and computations, we will consider the area illuminated by the radar beam to be a rectangle (Figure 5.1).



Example:

On this figure, parameters have the following values:

out_height = 4

scanned_width=20

bw_sample=5

Figure 5.1: The illuminated area is approximated by a rectangle.

Table 5.1: Definitions.

<i>Parameter</i>	<i>Notation</i>
Range index for the beam steps	y
Range index for the samples within the beam	sampley
Azimuth index for the beam steps	x
Azimuth index for the samples within the beam	samplex
Total number of range beam steps in the range width of the scanned area	out_height
Total number of azimuth beam step in the azimuth width of the scanned area	scanned_width
Total number of azimuth samples within the radar beam	bw_sample
Minimum starting y value for the next range step at a fixed x	start_index[x]
Flag that indicates when the point is out of the bin	flag_end
Complex weighted return electric field from a scatterer	point_signal
Sum of the complex weighted return electric fields (ideal approach only)	retsig
Sum of the complex weighted return electric fields at frequency n (real sensor only)	rtm[n]
Return power for the current beam (ideal approach only)	step_sum
Return power for the current beam indexed by x, at frequency n (real sensor only)	step_sum[x][n]
Number of frequencies of the system	n_total_freq
Shadowing parameter for the current small cell	shadow

The actual radar measurement is the returned electric field from a slice of terrain within two wave fronts. In Figure 5.2, for example, the radar will receive the returns from cells 2, 3, 4, 5, 6. Cell 6 is between the two wave fronts even though its abscissa is not within the range width of the beam. The radar will not receive any return from cell 1 since it is not within the wave fronts. This issue comes from the fact that the terrain is not flat. Furthermore, another problem may arise since we consider that a cell is in the range slice only if its center point is in it: if the small cells are not sufficiently small, the case where there is no center point of a cell between the two wave fronts may happen (for non-flat terrain only). This is the reason why we let these cells sizes be smaller than the range width of the radar beam.

Therefore, a special scanning process will be necessary to take care of these phenomena. The algorithm we will study in Section 5.2.2 solves these problems.

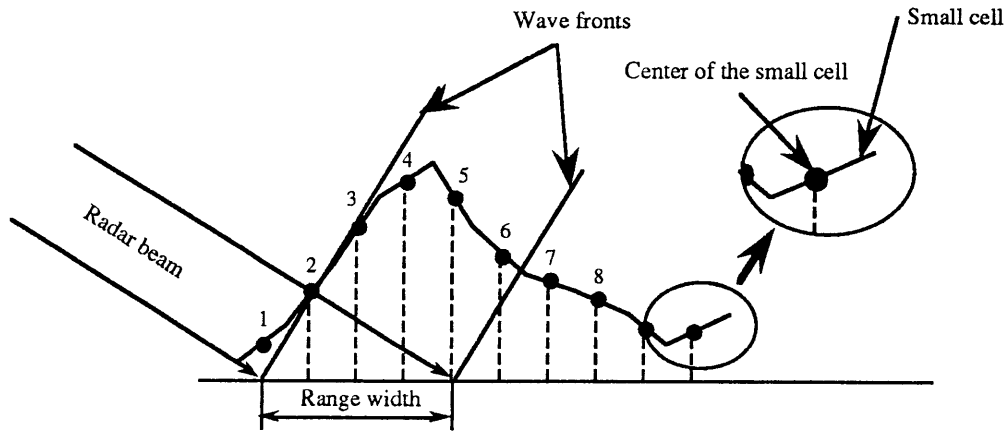


Figure 5.2: The radar return corresponds to the terrain between the two wave fronts.

5.2.2 Algorithms

One possible scanning algorithm is presented in Figure 5.3 using a Processing Logic Tree diagram. As it is stated in Section 3.2, we start the scan in the near left of the area. Then, we scan in x first. For each azimuthal step of the beam, we also scan in $samplex$ within the beam (discretization of the illuminated area, the step is also the azimuthal step). For each of these steps within the beam, we apply the following method:

- The sample index in the range direction ($sampley$) is set to the previously computed index of start ($start_index$) or to zero if this is the first row of scan.
- Then we scan in the range direction indexed by $sampley$ (the step is equal to the range width divided by the oversampling factor) until we find a scatterer between the two wave fronts (then, we set $flag_end$ to 1). When this scatterer is found, we will continue to scan in $sampley$ until we leave the layer between the two wave fronts. For each of the scatterers, we update the coherent sum of returned electric fields.
- When $flag_end$ is equal to 1, we set $start_index$ to the values $sampley$ for the next row and go to the next step in $samplex$.

When the area illuminated by a beam is scanned we compute the return power. When a row in x is scanned, we save it in a file. The content of this file will be converted into levels of gray to synthesize the return power image.

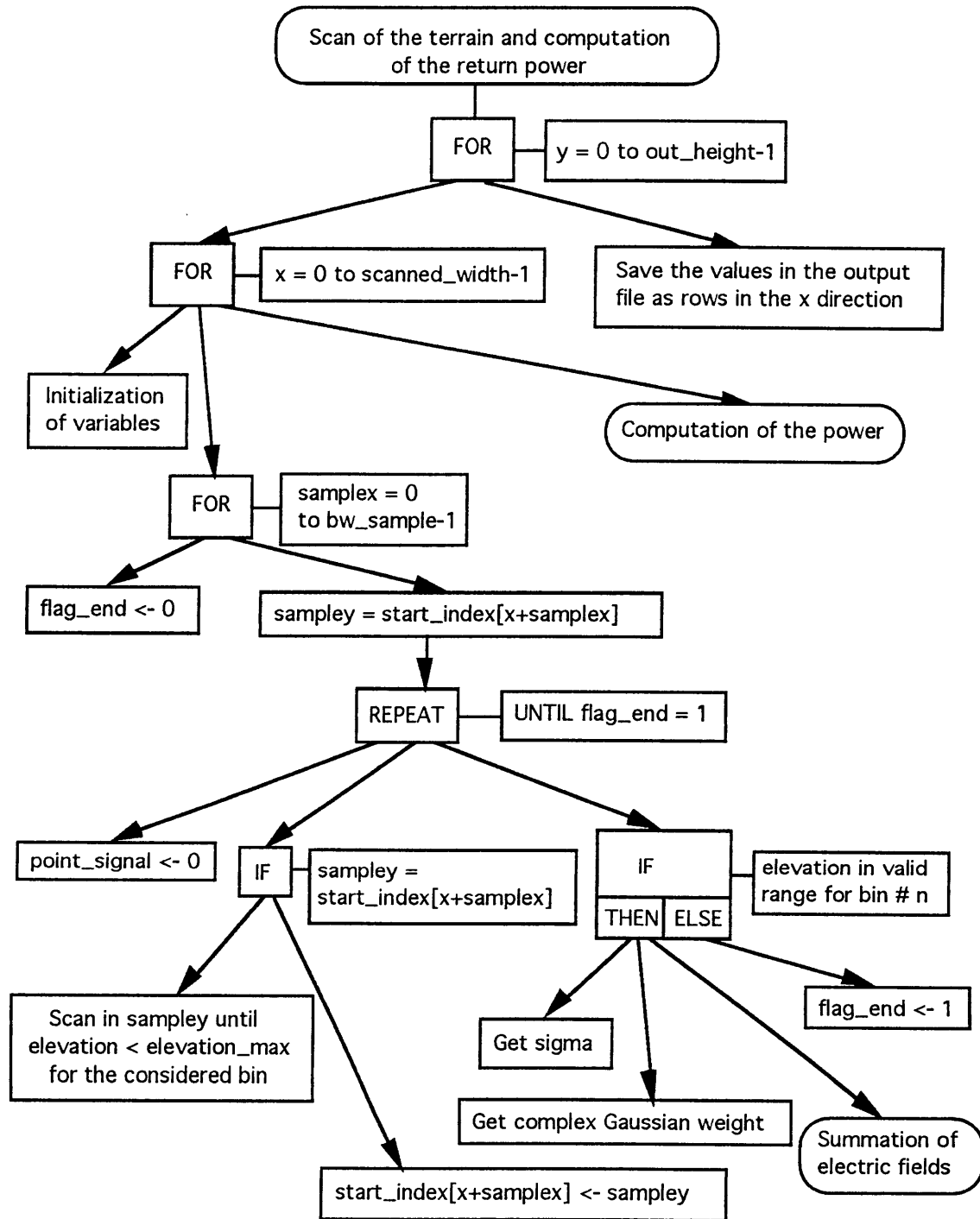


Figure 5.3: Scan of the terrain and computation of the return power.

This is the place where the idealized image synthesis and real sensor image synthesis algorithms part. First we will study the idealized approach. For the coherent summation of the returned electric fields, we need to take the shadowing effects into account. The algorithm presented in Figure 5.4 includes this feature. In this case, the computation of the power from the electric field is very simple (Figure 5.5).

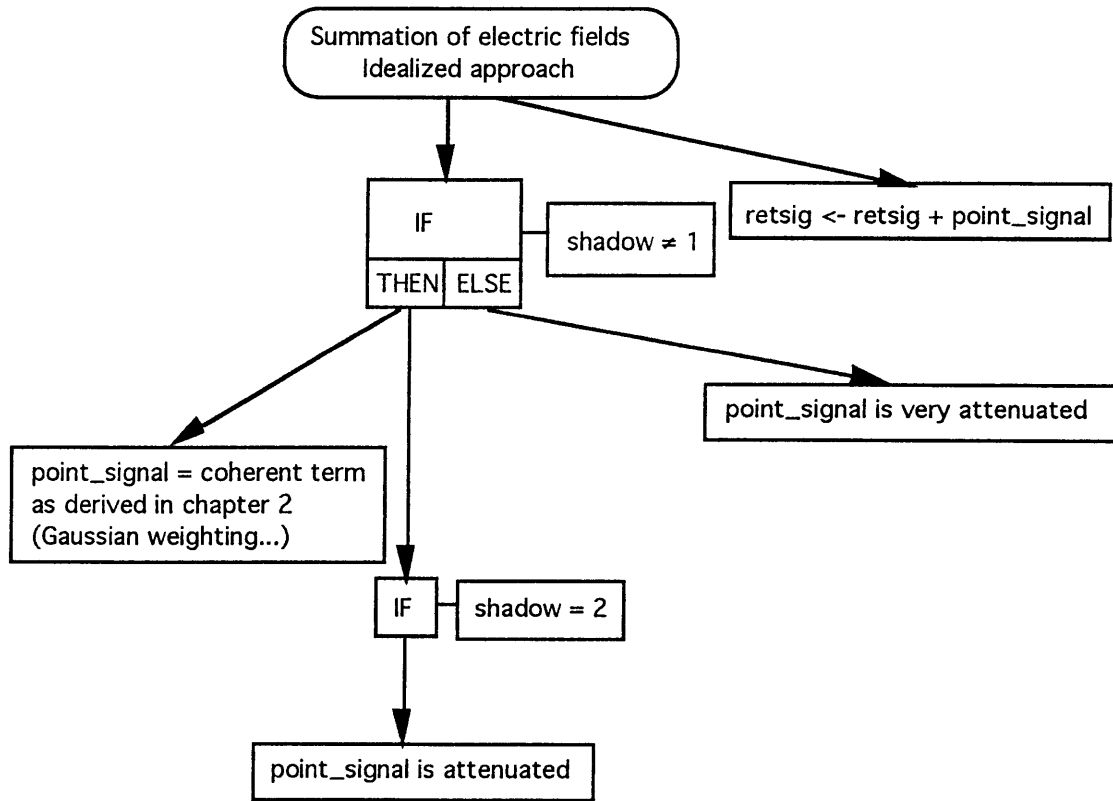


Figure 5.4: Summation of the electric fields, idealized approach.

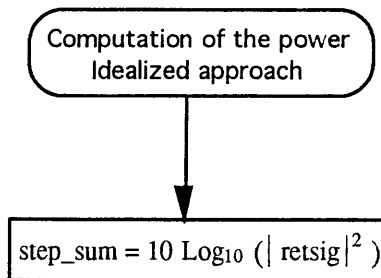


Figure 5.5: Computation of the power, idealized approach.

The real sensor case is more complicated since it has to include the conversion frequency data versus high resolution range data detailed in Section 3.2.2. The algorithms in Figures 5.6 and 5.7 use this conversion process. Instead of being computed at only one frequency as in the idealized case, the returned electric field is computed for each of the n_{total_freq} frequencies of the system at each step. These n_{total_freq} results are then converted using an inverse Fourier transform as is shown in Figure 5.7.

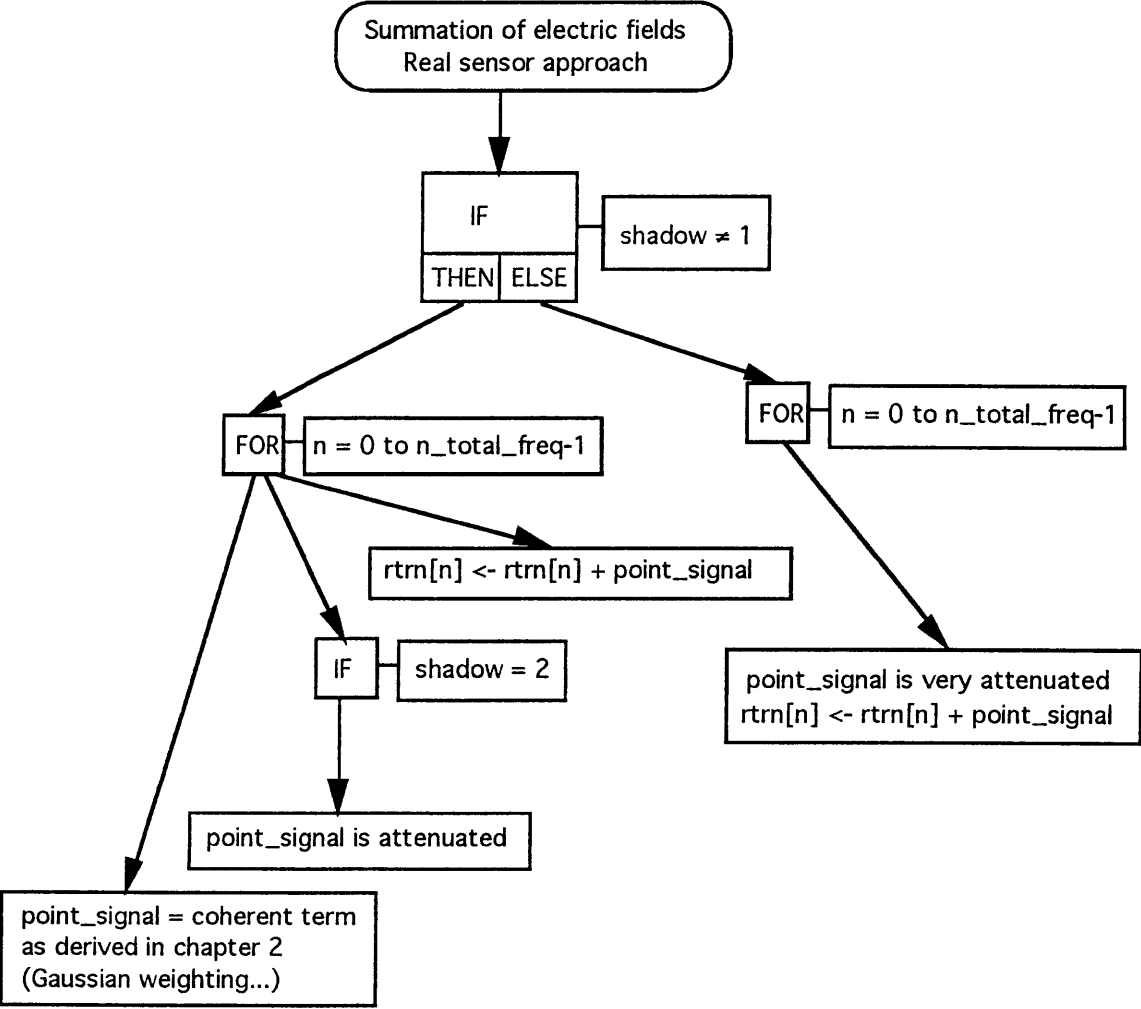


Figure 5.6: Summation of the electric fields, real sensor approach.

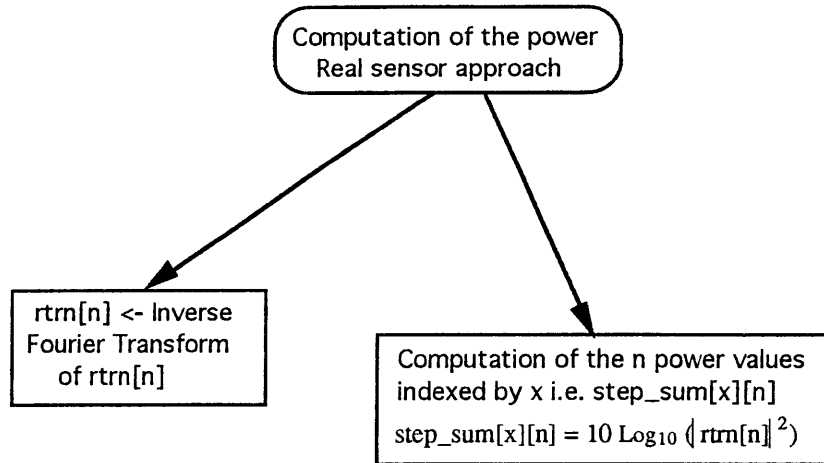


Figure 5.7: Computation of the power, real sensor approach.

This study shows that the program architecture can be identical for both idealized and realistic radar image synthesis. The only differences arise in the summation of the electric fields and in the computation of the return power.

These two algorithms are implemented in C programs.

Chapter 6

RESULTS, INTERPRETATION

6.1 Introduction

In the previous chapters, we investigated two software algorithms of stepped frequency radar image synthesis. For each of these two simulation approaches, we studied the architecture and performed the high level development of an application program. Meanwhile, two computer programs were implemented in C language on a Silicon Graphics Iris Indigo/R4000 workstation (listings in Appendix B and C), corresponding to the idealized image synthesis and the real sensor image simulation.

These programs were extensively tested and many simulations were done. As input terrain maps we used three versions of the same map, using the same elevation data. The first terrain map contains only vegetation. On the second one, a road and a lake were added. For the third one, we included an airport runway.

The radar we will simulate in this document is a stepped frequency system with a reference frequency of 35 GHz. This system has 32 frequency steps, 10 MHz apart.

First we will simulate the image of the runway using a flat elevation map for both idealized and real sensor synthesis. Then, for each three dimensional terrain region, both idealized and real sensor image synthesis will be completed.

For this computer simulation, we modeled several terrain types. The relationship between these terrain type and their gray level on the input maps is given in Table 6.1.

Table 6.1: Terrain types, characteristics, and corresponding gray level.

<i>Terrain type</i>	<i>Characteristics</i>	<i>Gray level</i>
Trees	23 meters high	white
Grass	0.3 meter high, low moisture	gray
Grass	0.3 meter high, more moisture	dark gray
Roads	asphalt	lightest gray
Lakes	N/A	black

The elevation data is saved as a binary file; however, in order to show it in this document, it was converted into an image with 256 gray levels. Low elevations are darker and high elevations are lighter. This elevation map is shown in Figure 6.1 (next page). The lowest elevation of the map is 366.3 meters and the highest is 423.6 meters.

For each simulation presented in this chapter, we will give the input map corresponding to the scanned region and a portion of the listing of the file, generated by the program, containing a summary of the input data. On each original terrain map, the imaged region has been surrounded by a white curve on the map. The areas corresponding to these scanned regions are shown on the elevation map (Figure 6.1).

We will present the results of both idealized and real sensor image synthesis algorithms. For these synthesized return power images the following definition was chosen: the lower the returned power, the darker the image area. Then, we will comment on these synthesized images.

Finally we will study some of the limitations of the simulation methods and the computer programs.

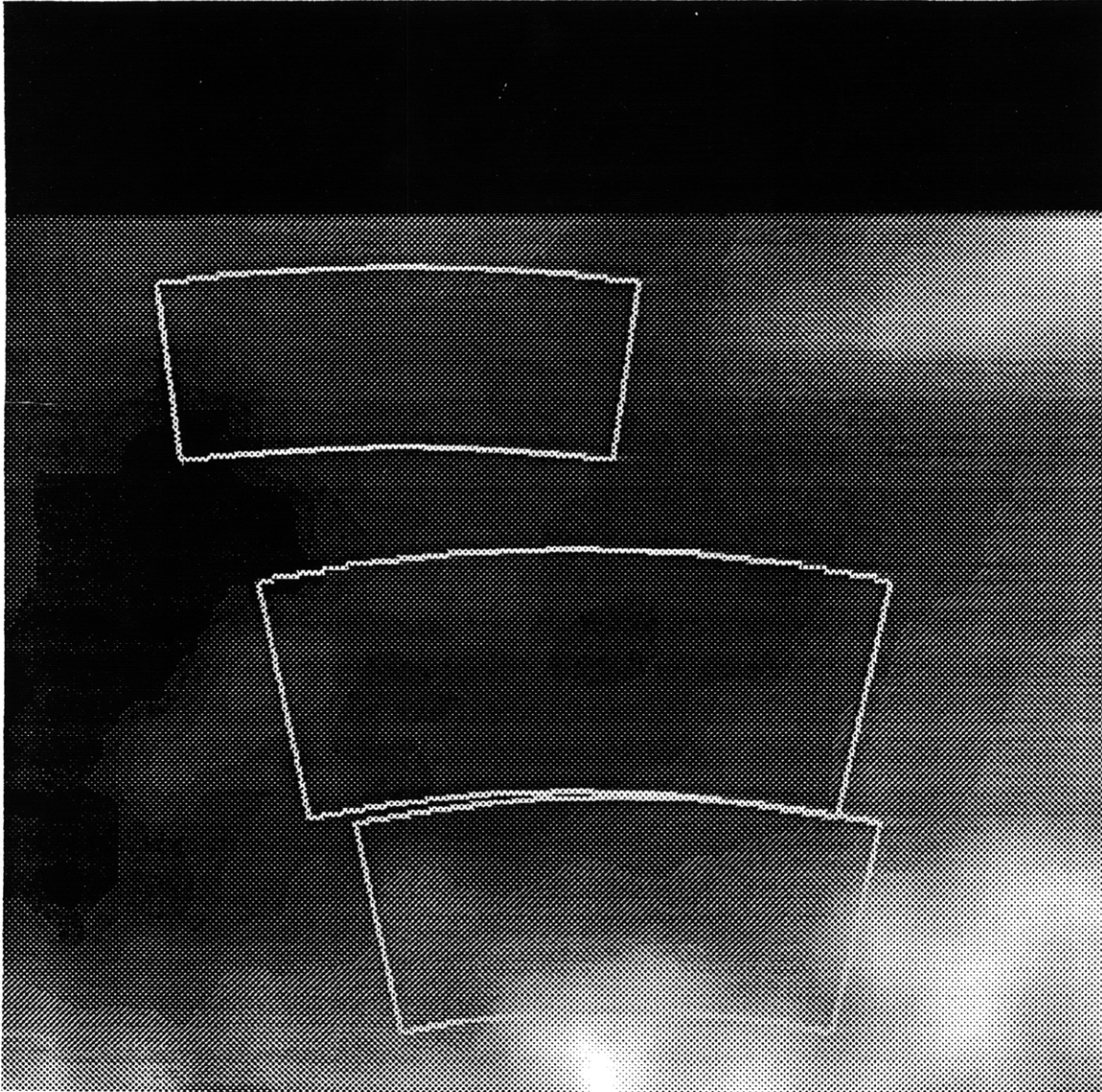


Figure 6.1: Elevation data map.

Scanned region for vegetation only: bottom area.

Scanned region for road, vegetation and lake: top area.

Scanned region for vegetation and runway: middle area.

6.2 Image Synthesis: Flat Terrain

In this section, we study the simulation of radar image applied to a flat map. The runway map is used as an example. The corresponding scanned area is presented in Figure 6.2.

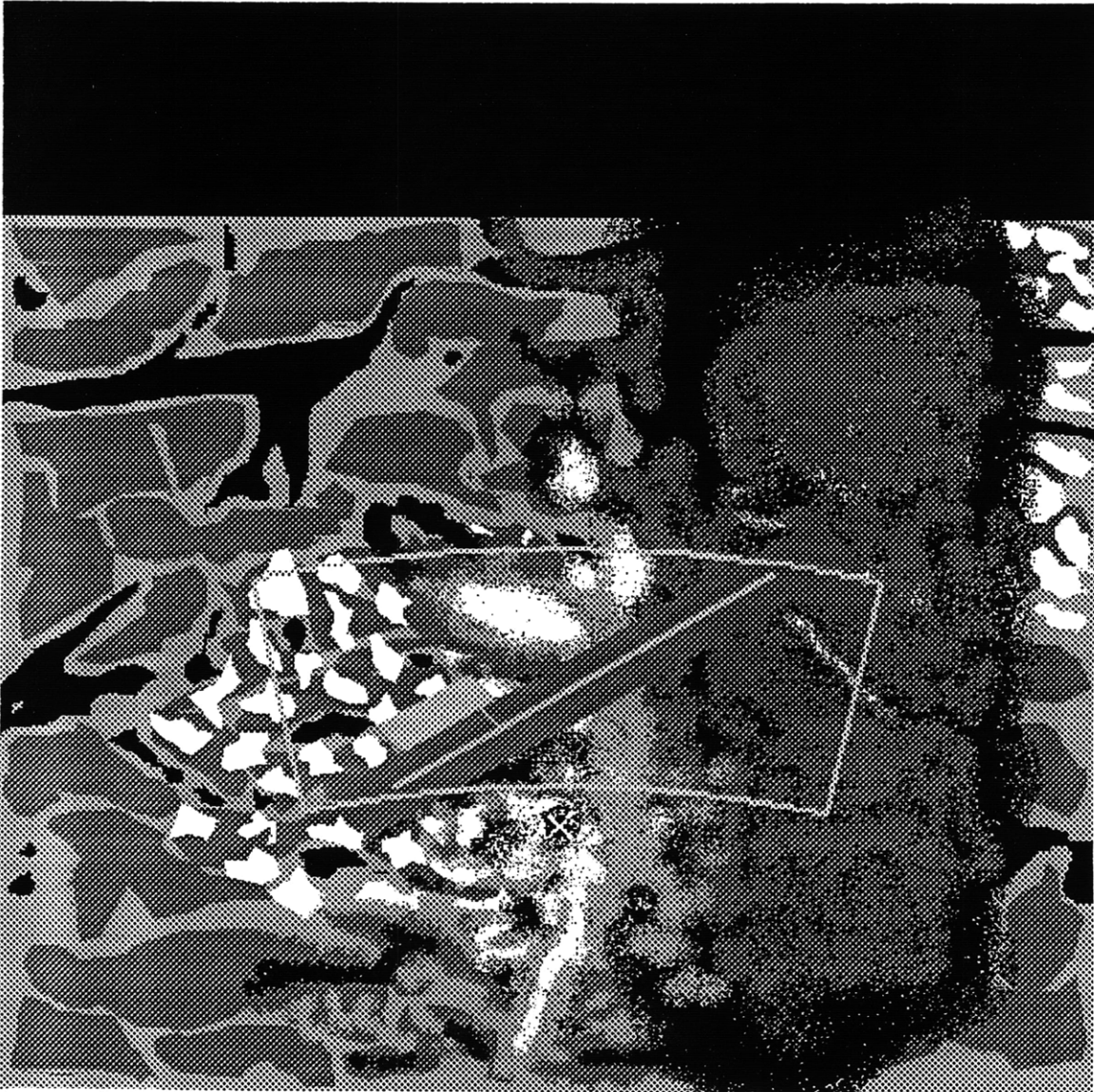


Figure 6.2: Scanned area for flat terrain simulation.

6.2.1 Idealized Approach

A summary of the configuration of the simulation is given in Table 6.2.

Table 6.2: Flat terrain radar image simulation characteristics.

```
*****
Sensor height = 720.0 m
Range length of area to be scanned = 270.0 m
Sensor distance to bottom left start of scan = 1350.0 m
Reference frequency = 35.0 GHz
Frequency step = 10.0 Mhz
Number of frequency steps = 32
Sensitivity = -50.000000 dB
Distance per range step = 0.42 m
Oversampling factor = 2 times
Azimuthal length of scan = 25.00 degrees
Azimuthal step size = 0.10 degrees
Beam width of sensor = 0.10 degrees
Sensor location [global x,y] = 650 , 0 meters
Sensor angle from north = 0 : 0.00 degrees
Global x,y location of terrain map origin = 0 , 1000 meters
*****
```

Figure 6.3 is the synthesized image of the area surrounding the runway.

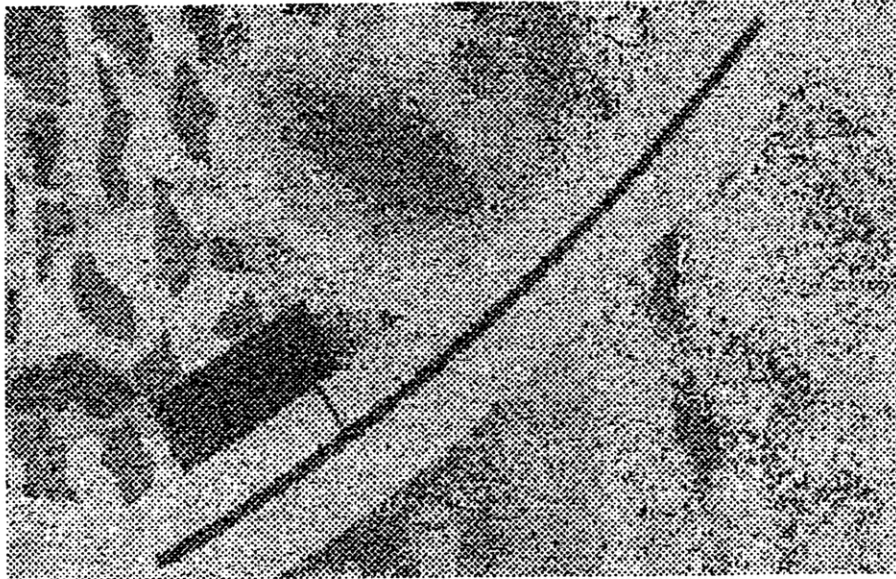


Figure 6.3: Synthesized image for flat terrain, idealized sensor simulation.

6.2.2 Real Sensor Approach

The simulation was done in the same conditions as for the idealized synthesis. The range width of the radar beam is not equal to the range resolution anymore but to 32 times (for 32 frequencies) its value. This will always be the case for real sensor simulation. Furthermore, the oversampling factor is equal to the number of frequencies (i.e. 32) in this case, instead of 1 for the idealized method. This difference comes from the fact that the oversampling factor is defined as the number of samples in a range width of the radar beam. An oversampling factor of 1 was chosen for the idealized simulation, therefore the equivalent oversampling factor is 32 for a 32 frequency real system simulation (the range width is 32 larger than in the idealized case). We will always use an oversampling of 32 in the real sensor simulations presented in this thesis.

The simulated radar image is shown in Figure 6.4.



Figure 6.4: Synthesized image for flat terrain, real sensor simulation.

6.2.3 Interpretation

The synthesized image is almost the same for both idealized and real sensor simulations. This result is in total accordance with the derivation of Section 3.2.2 where the idealized method was shown to be equivalent to the real sensor image synthesis for flat terrain case. The only differences between the two simulated images come from the limits of the calculation accuracy of the computer for the inverse Fourier transform that was used in the real sensor synthesis. If a Fast Fourier Transform (FFT) algorithm had been used to perform this computation, the calculation errors would have been even larger.

The distortion which appears on the simulated radar image is due to the synthesis algorithm. This phenomenon is one of the limitations of the computer programs and will be discussed in Section 6.6.

6.3 Image Synthesis: Vegetation

The studied input region contains only vegetation. The terrain is not considered to be flat anymore, the elevation data presented in Figure 6.1 is used. Figure 6.5 represents the map of the scanned area.

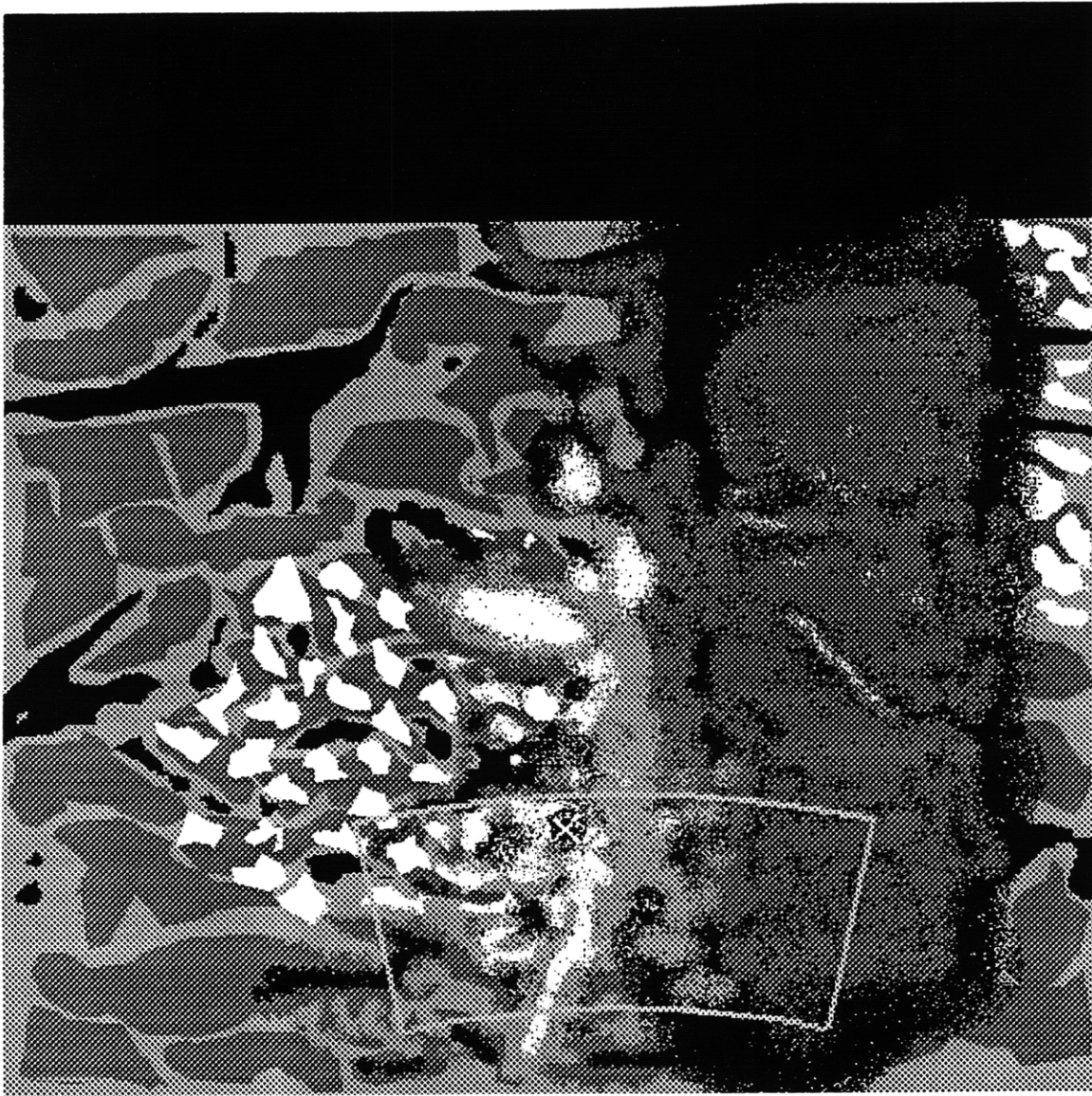


Figure 6.5: Scanned region, non-flat terrain, vegetation only.

6.3.1 Idealized Approach

The characteristics of the simulation are shown in Table 6.3. For this region, two simulation were performed. The only difference between these two synthesis was the value of the oversampling factor. This parameter was set to 1 for the first simulation and to 2 for the second one. We will comment on the influence of this factor.

Table 6.3: Non-flat vegetation terrain radar image simulation characteristics (vegetation area).

```
*****
Sensor height = 720.0 m
Range length of area to be scanned = 240.0 m
Sensor distance to bottom left start of scan = 1100.0 m
Reference frequency = 35.0 GHz
Frequency step = 10.0 Mhz
Number of frequency steps = 32
Sensitivity = -50.000000 dB
Distance per range step = 0.42 m
Oversampling factor = 2 times
Azimuthal length of scan = 25.00 degrees
Azimuthal step size = 0.10 degrees
Beam width of sensor = 0.10 degrees
Sensor location [global x,y] = 650 , 0 meters
Sensor angle from north = 0 : 0.00 degrees
Global x,y location of terrain map origin = 0 , 1000 meters
*****
```

Figures 6.6 and 6.7 present the synthesized image for an oversampling factor of 1 and 2, respectively.

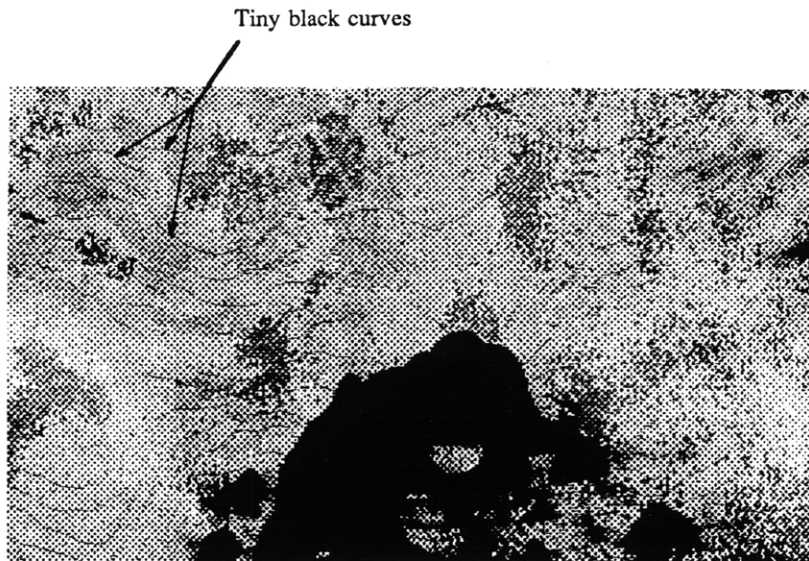


Figure 6.6: Synthesized image for non-flat terrain simulation (oversampling factor set to 1).

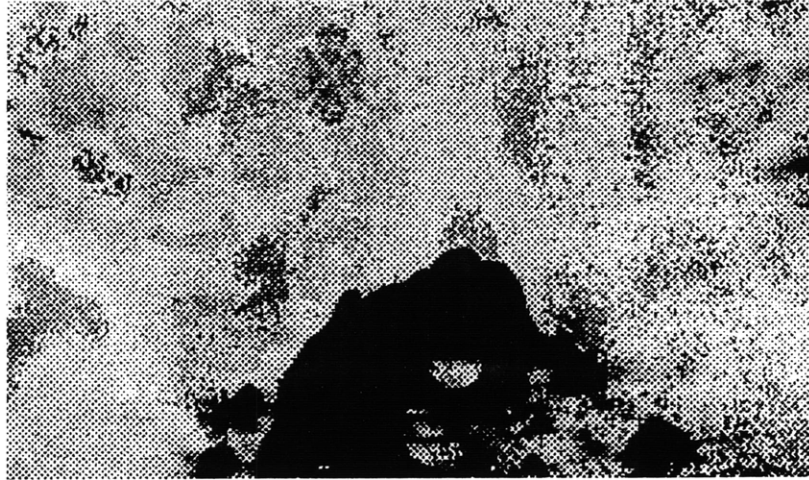


Figure 6.7: Synthesized image for non-flat terrain simulation (oversampling factor set to 2).

6.3.2 Interpretation

Several issues appear on the synthesized images. In the case where the oversampling factor is 1, very tiny black curves show up on the simulated image. This phenomenon does not exist on the other simulation that used an oversampling factor of 2. These black points are due to the terrain map sampling step. This issue was mentioned in Section 5.2.1 and occurs only for non-flat terrain. When the oversampling factor is not large enough, there may be no terrain small cell between the two wave fronts used for the computation of the return power and therefore the return power is null for the corresponding area. This is a pure computational problem which can be solved easily by using a larger oversampling parameter. Even with an oversampling factor of only 2 this phenomenon usually disappears. Therefore, for all the simulations we will perform, since the terrain is not flat, the oversampling parameter will be equal to 2. For steep terrain, it could be necessary to increase this parameter even more to get accurate results.

Another issue is due to the three dimensional aspect of the studied area. Figure 6.8 shows the elevation map corresponding to the area surrounding the imaged region (the darker the lower).

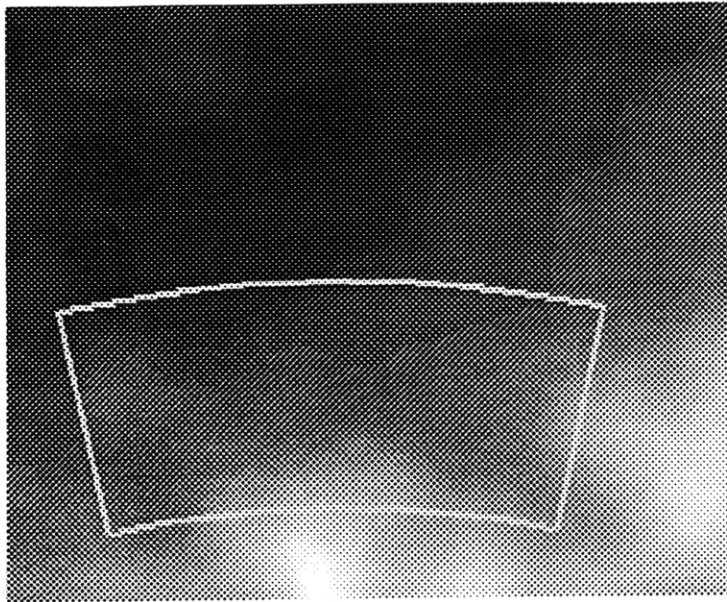


Figure 6.8: Elevation map corresponding to the imaged region.

The radar sensor is far from the imaged region and not high enough to prevent the scanned region from being partly in the shadow of the hills we can see in the foreground of Figure 6.8. The radar return of a shadowed region is greatly attenuated and therefore, the synthesized radar image is very dark where a shadow occur. This is why a large part of the simulated radar image is dark.

The comparison between the input terrain map and the synthesized radar image is fruitful. The shapes of the different terrain areas appears on the radar image. Water regions appear in dark on the radar image (low power return), forests are lighter. However it is very difficult to distinguish between the two types of grass. Since the radar return from a terrain depend on its orientation, its radar image changes with the terrain slope variations. Hence, the radar image of a non-flat terrain is farther from the real terrain map than for a flat terrain.

6.3.3 Real Sensor Approach

The simulation was done under the same conditions as for the idealized synthesis. However, the oversampling factor is equal to 32 (i.e. one point per range resolution bin). Figure 6.9 shows the synthesized image.

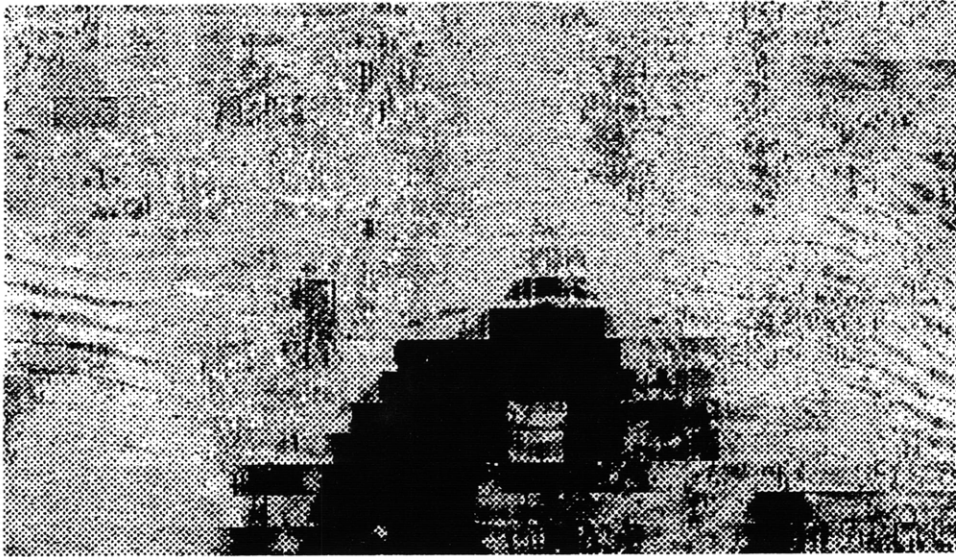


Figure 6.9: Synthesized image for non-flat terrain real sensor simulation (vegetation).

6.3.4 Interpretation

The retrieval of the high resolution radar image of a non-flat terrain in the real sensor case includes the use of an inverse Fourier transform. In Section 3.2.2, it was shown that the real sensor synthesis is not equivalent to the idealized approach. The conversion between several measurements of the returned electric field for the same area at different frequencies to returned powers from several range bins does not allow the exact recovery of the idealized high resolution terrain image. The real sensor radar image is distorted and noisy. Figure 6.9 is a good example of this distortion. The shapes of forests and small lakes are very noisy. A further digital image processing seen to be necessary to smooth this image. Without this processing it is very difficult to analyze this vegetation image.

6.4 Image Synthesis: Road, Vegetation, and Lake

In this section, we study the simulation of radar image applied to the non-flat map containing a road and a lake. The input terrain region is presented in Figure 6.10.

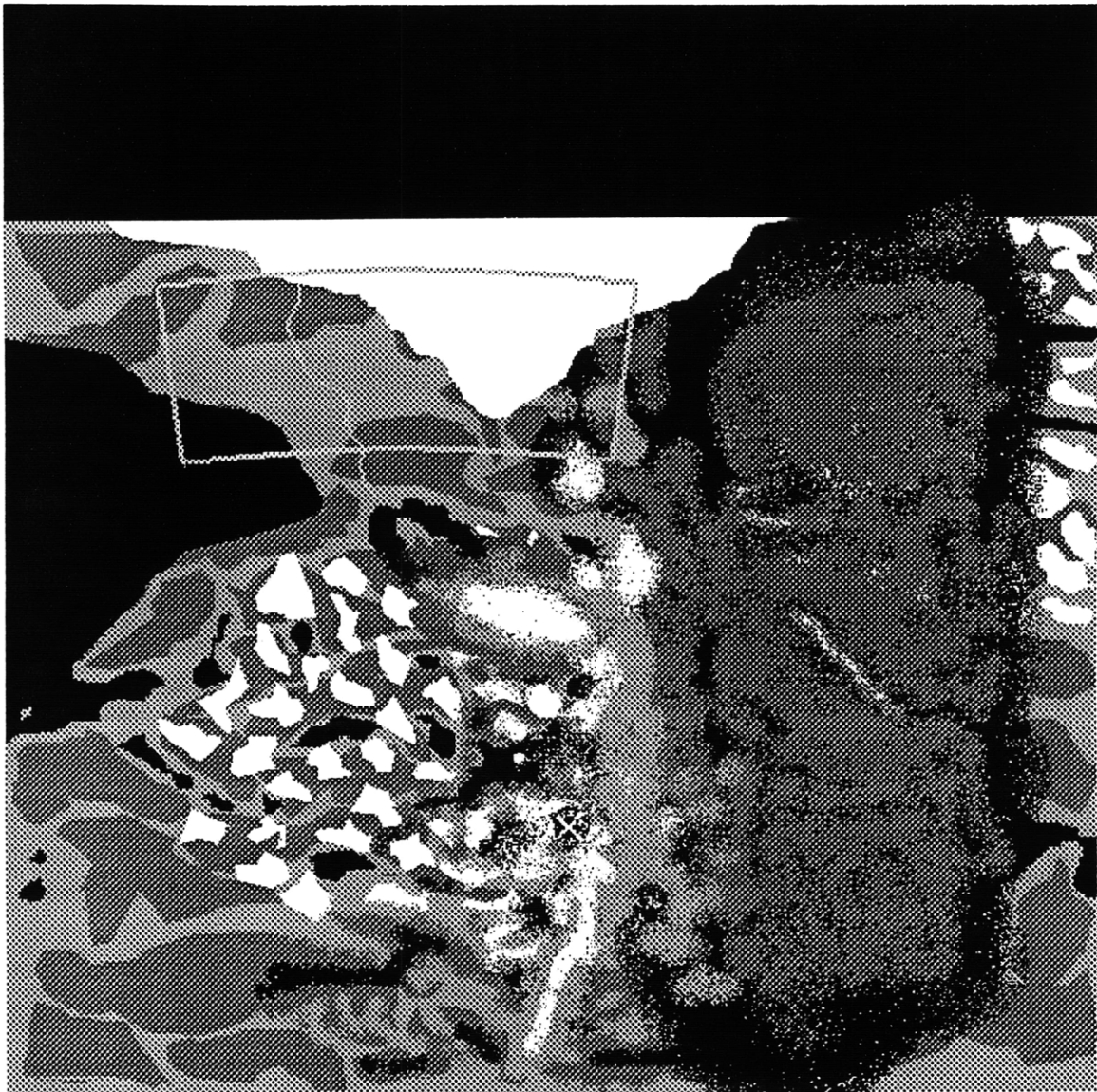


Figure 6.10: Scanned region, non-flat terrain (road, vegetation, and lake).

6.4.1 Idealized Approach

The simulation characteristics are summarized in Table 6.4.

Table 6.4: Non-flat terrain radar image simulation characteristics (road, lake, and vegetation).

```
*****
Sensor height = 720.0 m
Range length of area to be scanned = 200.0 m
Sensor distance to bottom left start of scan = 1740.0 m
Reference frequency = 35.0 GHz
Frequency step = 10.0 Mhz
Number of frequency steps = 32
Sensitivity = -50.000000 dB
Distance per range step = 0.42 m
Oversampling factor = 2 times
Azimuthal length of scan = 16.00 degrees
Azimuthal step size = 0.10 degrees
Beam width of sensor = 0.10 degrees
Sensor location [global x,y] = 650 , 0 meters
Sensor angle from north = 0 : 0.00 degrees
Global x,y location of terrain map origin = 0 , 1000 meters
*****
```

The oversampling factor is set to 2 to avoid the tiny black curves mentioned in Section 6.3.2, from showing up on the radar image simulation.

Figure 6.11 shows the synthesized image for the idealized sensor approach.

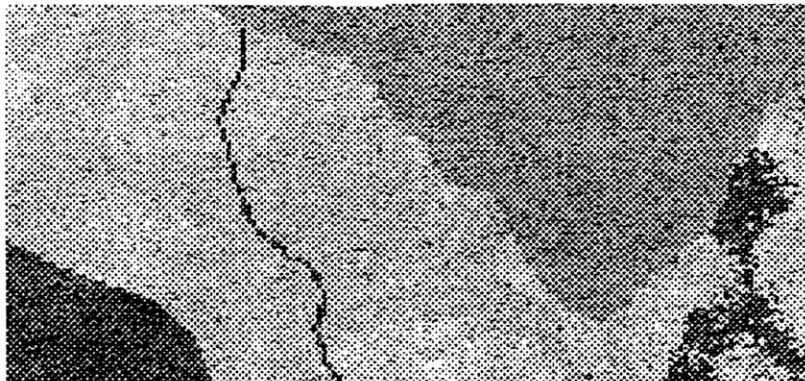


Figure 6.11: Synthesized image for non-flat terrain idealized simulation (road, lake, and vegetation).

6.4.2 Real Sensor Approach

Figure 6.12 shows the real sensor simulated image.

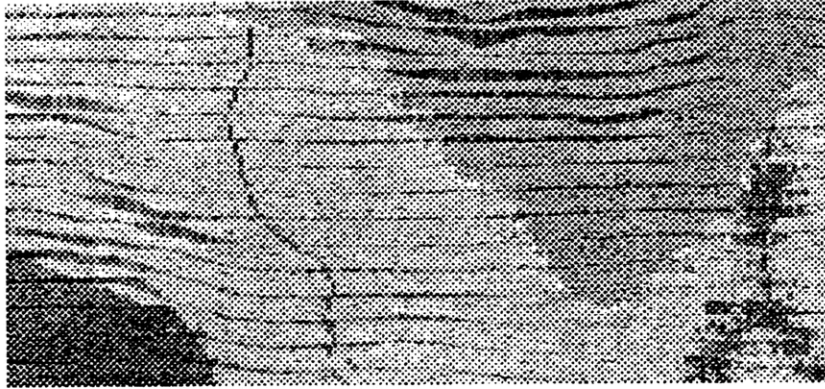


Figure 6.12: Synthesized image for non-flat terrain real sensor simulation (road, lake, and vegetation).

6.4.3 Interpretation

The synthesized image obtained by the idealized simulation is very close to the real terrain map even though the terrain vary in this region. The road appears clearly, lake and forest are unambiguously shown too.

The real sensor simulation result is much better than for the previous simulation (vegetation only). The road, lake and forest can still easily be recognized by a human eye. However, a computer analysis of this image may induce recognition errors (for the automatic update of an aircraft navigation system for example). The difference between this image and the one given by the idealized method is due to elevation variations. A special digital image processing may be useful.

6.5 Image Synthesis: Vegetation and Runway

We now study the radar image synthesis applied to the non-flat terrain surrounding a runway. The input terrain region is the same as in the flat terrain simulation (Figure 6.2).

6.5.1 Idealized Approach

Table 6.5 gives a summary of the characteristics of the simulation. The synthesized image is shown in Figure 6.13.

Table 6.4: Non-flat terrain radar image simulation characteristics (vegetation and runway).

```
*****  
Sensor height = 720.0 m  
Range length of area to be scanned = 270.0 m  
Sensor distance to bottom left start of scan = 1350.0 m  
Reference frequency = 35.0 GHz  
Frequency step = 10.0 Mhz  
Number of frequency steps = 32  
Sensitivity = -50.000000 dB  
Distance per range step = 0.42 m  
Oversampling factor = 2 times  
Azimuthal length of scan = 25.00 degrees  
Azimuthal step size = 0.10 degrees  
Beam width of sensor = 0.10 degrees  
Sensor location [global x,y] = 650 , 0 meters  
Sensor angle from north = 0 : 0.00 degrees  
Global x,y location of terrain map origin = 0 , 1000 meters  
*****
```



Figure 6.13: Synthesized image for non-flat terrain idealized simulation (vegetation and runway).

6.5.2 Real Sensor Approach

Figure 6.14 gives the resulting return power synthesized image.



Figure 6.14: Synthesized image for non-flat terrain real sensor simulation (vegetation and runway).

6.5.3 Interpretation

On both idealized sensor and real sensor images the identification of the runway and the parking are easy. Since the terrain is nearly flat in this area the real sensor image is relatively accurate about the contours of the forests and small lakes. However there is still some noise due to the imperfections of the high resolution image recovery.

6.6 Limitations

The limitations we will study are due either to the radar scattering model employed or to the image simulation method.

6.6.1 Image Distortion

Two different types of distortion occur in the image synthesis that were performed in this thesis. The first one is due to the simulation method. The actual imaged area is not a rectangle but a slice of crown and the synthesized image is a rectangle, therefore it is slightly distorted (Figure 6.15).

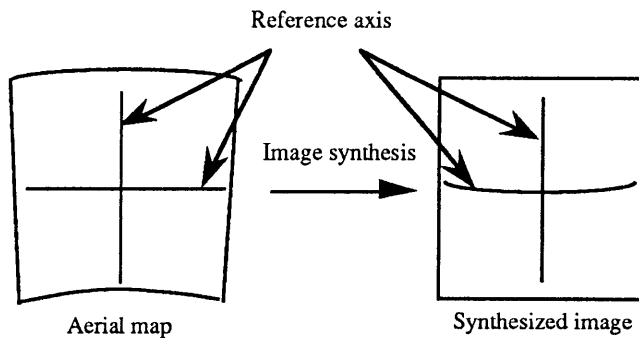


Figure 6.15: Image distortion.

This distortion is deterministic (it only depends on the azimuthal and range widths of the imaged area) and can be easily corrected by an adequate image processing.

The second type of distortion was encountered for the real sensor image synthesis. This distortion is due to the high resolution image recovery method. Its amplitude directly depends on the elevation variations of the imaged region. Therefore, it can not easily be compensated. A special image processing technique has to be developed for this purpose.

6.6.2 Scattering Model Limitations

Since we will deal with three dimensional terrain, some approximations were necessary, especially for trees. Actually, the E.M.S.A.R.S. model provides us with backscattering coefficients as a function of incidence α with respect to the vertical (Figure 6.16).

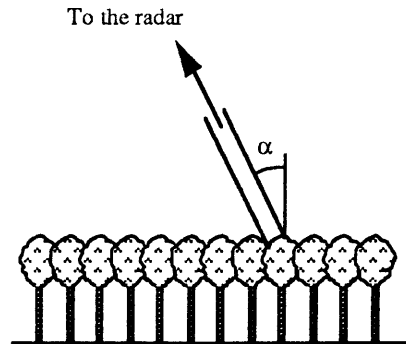


Figure 6.16: Backscattering coefficients are a function of the orientation α of the radar line of sight to the terrain.

The point is that this approach is theoretically only valid for an horizontal terrain. When the terrain is tilted, the trunks of the trees are not perpendicular to the ground anymore but they are still very much vertical (Figure 6.17). Therefore, it should not be sufficient to consider that the backscattering properties of trees only depend on the angle α (Figure 6.16), these properties should also depend on β , the terrain inclination (Figure 6.17).

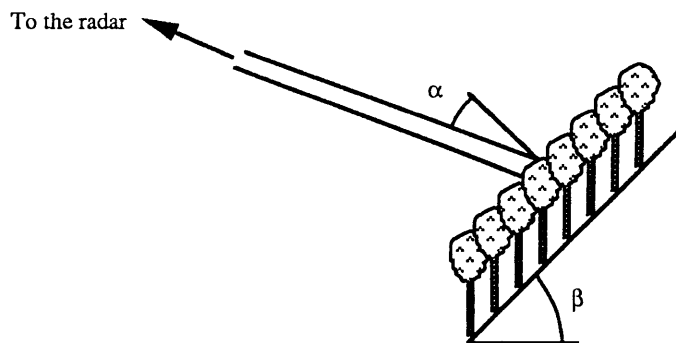


Figure 6.17: The backscattering properties of trees depend on both α and β .

In order to evaluate the amplitude of this phenomenon, several backscattering property comparisons were made between the case where we only use α and the one in which both α and β are taken into account.

The conclusion of this study was that the difference between the two results was very small except when the radar was illuminating the tree line at low elevation angles (this comes from the fact that leaves usually scatter more than trunks). For this reason, the simulation programs take only the angle α into account which provides a good approximation of the real backscattering coefficients.

The radar return of a forest is much larger at its foreground edge (tree line) than everywhere else. This phenomenon was observed on experimental data. There is no model to deal with this raised power, therefore we used the results for trees, raised by 20 dB (this value corresponds to experimental data), for tree lines.

In this simulation, we modeled the backscattering properties for road and water using the small perturbation method. This method only gives accurate results for such terrain types when the surfaces have small variations. Therefore, we have to consider that the lake surface is flat i.e. that there is no wind and that the road surface is very smooth.

6.6.3 Shadowing Effect Approximation

We made two minor approximations in dealing with the shadowing problem. First, we considered that a terrain scatterer is either in a shadow or not. In reality, terrain scatterers may be partially shadowed. However, since these scatterers are small compared to the range width of the radar beam, this approximation does not modify significantly the return power image.

Second, in the computation of the shadowing parameter of the small cells (Chapter 4), we implicitly assumed that the distance between two consecutive points (which is parametered by the oversampling factor) is sufficiently small to consider that we can neglect the shadowing effect due to a local maximum of the terrain between two consecutive points (Figure 6.18 (a) and (b)).

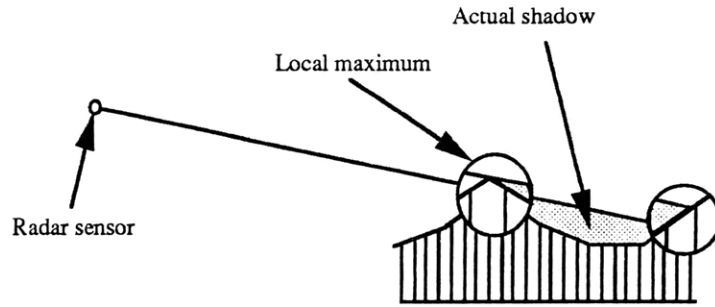


Figure 6.18 (a): Actual shadowing effect, the local maximum is taken into account.

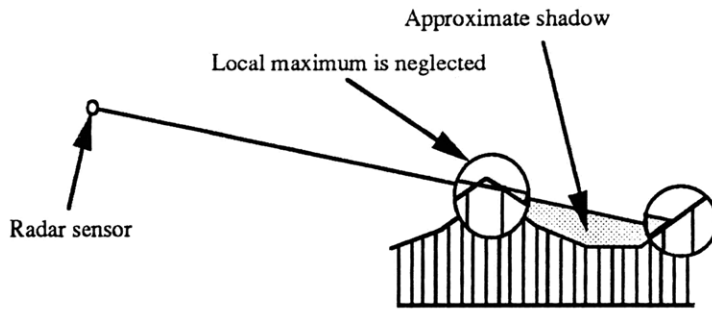


Figure 6.18 (b): The local maximum between two points is neglected.

In practice, even if the oversampling factor is only one or two, this approximation is totally valid since the range resolution of the radar is usually smaller than or equal to the resolution of the input terrain map.

Chapter 7

SUMMARY, CONCLUSIONS, AND SUGGESTED FUTURE WORK

7.1 Summary and Conclusions

In this thesis, a radar image simulation scheme has been developed. The following domains were investigated:

- Terrain backscattering properties modeling for both volume and surface scattering.
- Radar sensor modeling (return power to the radar).
- Radar image simulation techniques, including an idealized approach and a more realistic method.
- Development of two application programs based on the previous concepts.
- Synthesis of simulated radar images using the computer programs.

Theoretical surface and volume scattering models describe the properties of media in terms of a few parameters. A two-layer anisotropic random medium model program from MIT's E.M.S.A.R.S. (Electromagnetic Model of Scattering Applied to Remote Sensing) was used to account for the volume scattering properties of the imaged terrain. The surface scattering characteristics were calculated using the small perturbation method.

In order to take into account the speckle that appear on radar images, the radar sensor statistics was modeled. For this purpose, the coherent summation of the returned electric fields from the imaged area was performed using complex Gaussian weighting.

Two simulation techniques were investigated. First, we studied an idealized method based on the assumption that the range width of the ideal radar beam equals the actual radar effective range resolution. The second method uses a real radar simulation. The expression of the high resolution power pattern was derived for both cases. While this derivation comes directly from the expression of the returned electric field to the radar in the idealized case, it involves an inverse Fourier transform conversion in the real sensor case. It was shown that the idealized radar is equivalent to the ideal radar for flat terrain only.

The architectures of two computer programs based on these two simulation techniques were developed. Then, a possible implementation technique was studied in detail. It was shown that a pre-processing of the input terrain map is a necessary step. A possible radar scanning algorithm was designed to deal with the shadowing of some areas due to the three dimensional aspect of the terrain. This algorithm includes the recovery of the high resolution power image. These algorithms were used to implement the two application programs corresponding to the idealized and real sensor image synthesis, respectively.

In Chapter 6, several images synthesized by the application programs were presented. These images were selected to show all the issues that occurred during the numerous simulations performed for this research. First, a flat terrain case was studied, the results were in total accordance with the derivation presented in Chapter 3: both idealized and real sensor imaging methods give the same results for flat terrain. Second, we imaged a partly shadowed non-flat region with many small forests and lakes. The shape of these terrain clearly appear in the idealized simulation image. However, the real sensor image is very noisy. In another simulation, a road was shown to be easily recognizable even in the real sensor case. The results of these simulations were discussed and interpreted.

Lastly, we commented on some limitations of the models and simulation techniques used in this research. A further digital image processing was shown to be a possible extension of this study.

7.2 Suggested Future Work

The backscattering coefficients used to characterize the terrain were obtained using theoretical models. These models are idealized and may not be compatible with optimal image simulation. Therefore, a possible future work would be to improve these theoretical models and to develop a model to take care of tree lines.

In the simulations conducted in this thesis, we always assumed that the radar waves were only HH polarized. The techniques developed in this study can also be applied to HV and VV polarizations. More realistic simulation results may be obtained using a fully polarimetric approach.

As it was mentioned in the interpretation of the synthesized images, it may be useful to apply a further digital image processing to the simulation images. This digital processing would have two objectives. The first one would be to correct the image distortion caused by the fact that the imaged area is not rectangular. The second one would be to smooth the ideal sensor simulation images to reduce the noise due to the inverse Fourier transform based high resolution image recovery. These two image processing would be a necessary step to allow the use of such radar images as landing aid on an aircraft at night or in the fog.

Appendix A

THREE DIMENSIONAL GEOMETRY DERIVATION

Given two points M_1 and M_2 in a three dimensional coordinate system defined by $M_1(x_1, y_1, z_1)$ and $M_2(x_2, y_2, z_2)$ with either $x_1 = x_2$ or (exclusive) $y_1 = y_2$.

Given a vertical plane V defined by: if $M(x, y, z)$ is a point on V then $a x + b y = c$ (no z dependence since V is vertical).

Problem:

Find the coordinates of the intersection point between the line L which goes through M_1 and M_2 , and V.

Solution:

Let $M(x, y, z)$ be a point on L. We have $\overrightarrow{M_1M} = \begin{pmatrix} x - x_1 \\ y - y_1 \\ z - z_1 \end{pmatrix}$ and $\overrightarrow{M_2M} = \begin{pmatrix} x - x_2 \\ y - y_2 \\ z - z_2 \end{pmatrix}$

The vectors $\overrightarrow{M_1M}$ and $\overrightarrow{M_2M}$ are linearly dependent, thus we can write $\overrightarrow{M_1M} \times \overrightarrow{M_2M} = \vec{0}$ (cross product)

$$\text{i.e.} \quad \begin{pmatrix} x - x_1 \\ y - y_1 \\ z - z_1 \end{pmatrix} \times \begin{pmatrix} x - x_2 \\ y - y_2 \\ z - z_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (\text{A.1})$$

$$\text{i.e.} \quad (y - y_1)(z - z_2) - (z - z_1)(y - y_2) = 0 \quad (\text{A.2})$$

$$(z - z_1)(x - x_2) - (x - x_1)(z - z_2) = 0 \quad (\text{A.3})$$

$$(x - x_1)(y - y_2) - (y - y_1)(x - x_2) = 0 \quad (\text{A.4})$$

Since we want to find the intersection between L and V, we also assume that M is on V i.e.

$$a x + b y = c \quad (\text{A.5})$$

We can now study the two given cases separately:

a) Case $x_1=x_2$

We have obviously $x = x_1 (=x_2)$ and only the equations (A.2), (A.5) are useful

$$\text{i.e.} \quad y(z_1 - z_2) - z(y_1 - y_2) = y_2 z_1 - y_1 z_2 \quad (\text{A.6})$$

$$a x_1 + b y = c \quad (\text{A.7})$$

$$\text{so} \quad x = x_1 (=x_2) \quad (\text{A.8})$$

$$y = (c - a x_1) / b \quad (\text{A.9})$$

$$z = [(z_1 - z_2)(c - a x_1) / b - (y_2 z_1 - y_1 z_2)] / (y_1 - y_2) \quad (\text{A.10})$$

b) Case $y_1=y_2$

We have obviously $y = y_1 (=y_2)$ and only the equations (A.3), (A.5) are useful

$$\text{i.e.} \quad z(x_1 - x_2) - x(z_1 - z_2) = x_1 z_2 - x_2 z_1 \quad (\text{A.11})$$

$$a x + b y_1 = c \quad (\text{A.12})$$

$$\text{so} \quad x = (c - b y_1) / a \quad (\text{A.13})$$

$$y = y_1 (=y_2) \quad (\text{A.14})$$

$$z = [(z_1 - z_2)(c - b y_1) / a + (x_1 z_2 - x_2 z_1)] / (x_1 - x_2) \quad (\text{A.15})$$

Application:

The results obtained above will be used in the computation of the characteristics of the small cells in the pre-processing of the input data.

Appendix B

C CODE FOR IDEALIZED RADAR IMAGE SYNTHESIS

```

/*****
/*
/*          SIMULATION OF MILLIMETER WAVE RADAR          */
/*
/*          RETURN FROM A THREE DIMENSIONAL ENVIRONMENTAL SCENE          */
/*
/*          Idealized Sensor Simulation          */
/*
/*
/*          Created by          */
/*
/*          Philippe BERISSET          */
/*
/*          MASTER of SCIENCE thesis, June 1993          */
/*
/*          Massachusetts Institute of Technology          */
/*
/*          Copyright Philippe BERISSET, 1993. All rights reserved.          */
/*
*****/
/*
/* This program computes the simulated return image from a three          */
/* dimensional area using Gaussian distribution and Complex weighting          */
/*
*****/
/*
/* To compile this program, type the following command:          */
/*
/*cc IdealSensor.c -o Ideal complex.o ran1.o -limage -lgutil -lgls -lm -g*/
/*
*****/
/*

```

INSTRUCTIONS FOR USING THE SIMULATION PROGRAM:

Executable code: Ideal
Library files: complex.h
Additional files: ran1.o, complex.o

Usage: Ideal inputfile_name (example: Ideal in-road.ide)

Input file format: Separator is a space

Line 1:

 Radar reference frequency (in Ghz)
 Frequency step (in MHz)
 Number of frequency steps
 Sensor sensitivity (in negative power of 10 ie 5 means 0.00001)

Line 2:

 Radar elevation (in meters)
 Horizontal distance radar - area to scan (in meters)
 Over sampling factor (integer) for the pre-processing of the data

Line 3:

 Range length of area to scan (in meters)
 Azimuthal angle (in degrees) of full scan area
 Azimuthal step size (in degrees)
 Azimuth beam width (in degrees)

Line 4:

 Terrain map filename
 Terrain elevation data filename
 Terrain data image height (pixels)
 Terrain data image width (pixels)
 Azimuthal scale (meters/pixel)
 Range scale (meters/pixels)

Line 5:

 Sensor location: global x coordinate (m)
 Sensor location: global y coordinate (m)
 Angle from north = 0, increasing in clockwise direction (deg)
 Global x location of terrain map's origin (m)
 Global y location of terrain map's origin (m)

Line 6:

 Name of graphic output file (.sgi)
 Magnification of output (integer)
 Magnification of output (integer)

Input Files Used by Program:

- 1) input file as described above
- 2) backscattering data file, format:
 elevation angle backscattering_coef (dB)
- 3) filename.dat contains names of data files corresponding to the terrain types of the input file, format:
 terrain number(integer) filename of terrain data file
- 4) files named in filename.dat (terrain data files)

Output files: scanned area contour on the input map is in area.sgi
numerical output is in profile.out
profile.res contains a summary of inputs and outputs
hist.dat contains the histogram of output values

/*****/

```

/*****
/*
/*          THE CODE SECTION STARTS HERE          */
/*
/*
/*****

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "complex.h"
#include <gl/gl.h>
#include <gl/device.h>
#include <gl/image.h>

#define PI 3.141592653589793
#define RAD 57.2957795132    /* deg/radian */
#define MAX_ARRAYX 300
#define MAX_ARRAYY 4000
#define MAX_ARRAY 700
#define MAX_INT 10000
#define MAX_TERRAIN TYPE 10
#define ATT_SHAD2 10e-2
#define ATT_SHAD1 10e-5
#define TREE_LINE_THICKNESS 3 /* Thickness of the radar echo
                               of a treeline in meters*/

/*****
/*          STRUCTURES DEFINITION          */
/*****

struct cell{
    float slope;
    short int shadow;
    float hght;
    int ter;
};

struct point_coord{
    float x;
    float y;
    float z;
};

struct corner_struct{
    float x;
    float y;
    float z;
    int ter;
};

/*****
/*          FUNCTIONS DECLARATION          */
/*****

float  compute_dist();
float  get_sigma();
float  set_random_array();
void   draw_a_point();
void   first_scanning_process();
void   initializations();
void   open_summary_file();
void   read_input_data();
void   set_weight_array();

```

```

/*****
/*   GLOBAL VARIABLES DECLARATION   */
*****/

/* Graphics variables */
short   blackvec[3] = {0, 0, 0};
short   rgbvector[3];

/* File pointers */
/*register FILE *outip; to use for binary format */
FILE     *res;

/* Input data variables */
float    angle;
float    az_step_size;
float    beam_width;
float    data[10][30][2];
float    dfreq;
float    d_step_size;
float    elevation[MAX_ARRAY][MAX_ARRAY];
char     elevation_file[12];
float    freq;
float    l_scan;
int      n_total_freq;
char     outfilename[12];
int      over_sampling;
float    phi;
float    range;
float    sensitivity;
int      terrain[MAX_ARRAY][MAX_ARRAY];
char     terrain_file[12];
int      xmagnification;
int      xS;
int      xsize;
int      xT;
float    xyscale;
int      ymagnification;
int      yS;
int      ysize;
int      yT;
float    zS;
float    zscale;

/* Internal variables */
float    *dbuf;
float    dist;
float    *dptr;
float    d_mean;
float    d_near;
float    hghtmin;
float    hghtmax;
float    min;
float    minimum;
float    max;
float    maximum;
int      num[MAX_TERRAIN_TYPE];
int      out_height;
int      out_width;
int      scanned_width;
int      start_index[MAX_ARRAY];
struct cell ter_cell[MAX_ARRAYX][MAX_ARRAY];
float    weight_array[MAX_ARRAYX][MAX_ARRAY][2];
int      x_cmin;
int      x_cmax;

```

```

int          y_cmin;
int          y_cmax;
int          y_max;

/*****
/*          MAIN SECTION          */
*****/

main(argc, argv)
int argc;
char **argv;
{
/* Check the command entered */
if (argc != 2) {
    fprintf(stderr, "Usage: Ideal inputfile\n");
    exit(1);
}

    printf("\nProgram running, please wait...\n");

/* Read the input data */
    read_input_data(argv[1]);

/* Initializes the arrays */
    initializations();

/* Open the summary file and write in it */
    open_summary_file();

/* Read in terrain data file to terrain[] [] */
    load_terrain_map(terrain_file);

/* Fill sigma arrays with rcs versus angle for each terrain type */
    fill_sigma_arrays();

/* Pre-processing of the input data */
    first_scanning_process();

/* Set up array of weighting values */
    set_weight_array();

/* Coherent summation of the returns */
    coherent_summation();

/* Create the output files and the image */
    create_output();

    printf("\nImage synthesis completed.\n");

/* Display the internal steps */
/*  display_windows();
    sleep(50);
    gexit();*/
} /* end main */

```



```

/*****
/*          FUNCTIONS          */
*****/

/*****
float compute_dist(p1,p2)
*****/
struct point_coord p1,p2;
{
    float  d1,d2,d;

    d1 = p1.x-p2.x;
    d2 = p1.y-p2.y;
    d=sqrt(pow(d1,2)+pow(d2,2));
    return d;
}/* end compute_dist */

/*****
float set_random_array(idum)
*****/
int *idum;
{
    static int iset=0;
    static float gset;
    float fac,r,v1,v2;
    float ranl();

    if (iset == 0) {
        do {
            v1=2.0*ranl(idum)-1.0;
            v2=2.0*ranl(idum)-1.0;
            r=v1*v1+v2*v2;
        } while (r >= 1.0);
        fac=sqrt(-2.0*log(r)/r);
        gset=v1*fac;
        iset=1;
        return v2*fac;
    } else {
        iset=0;
        return gset;
    }
}/* end set_random_array */

/*****
load_terrain_map(terrain_file)
*****/
char terrain_file[];
{
    int val, type, m, terrain_temp[MAX_ARRAY][MAX_ARRAY];
    int n, grey[20];
    int terr_type[20], ttt, tp, gry;
    short rowbuf[4096];
    unsigned char charbuf[4096], *cptr, *st;
    register IMAGE *image,*outip;
    register int y;
    FILE *t_file,*e_file;
    long int format;

    if( (image = icopen(terrain_file, "r")) == NULL ) {
        fprintf(stderr, "Real: can't open input file %s\n", terrain_file);
        exit(1);
    }

```

```

    }
    if( (t_file = fopen("type_file.dat", "r")) == NULL ) {
        fprintf(stderr, "Real: can't open input file %s\n", t_file);
        exit(1);
    }

    ttt = 0;
    while(fscanf(t_file, "%d %d", &gry, &tp) != EOF) {
        grey[ttt] = gry;
        terr_type[ttt] = tp;
        ttt++;
    }
    xsize = image->xsize;
    ysize = image->ysize;

    st = &charbuf[0];          /*set pointer to first element */
    cptr = st;
    for(y=0; y<ysize; y++) {
        getrow(image, rowbuf, y, 0);
        stoc(rowbuf, charbuf, xsize);
        cptr = charbuf;
        for(m=0; m<xsize; m++) {
            val = *cptr;
            for(n=0; n<ttt; n++)
                if (val == grey[n]) type = terr_type[n];
            terrain[m][y] = type;
            terrain_temp[m][y] = val;
            cptr++;
        } /* end for */
    } /* end for */

    /* Draw the scanned area on the input map */
    for (m=0; m<3; m++) {
        for (y=(int) (phi*100); y<(int) (phi*100)+1; y++) {
            terrain_temp[ (int) ((xS-xT)/2+(range/2-m)*y/200/RAD) ]
                [ (int) ((yS-yT)/2+(range/2-m)*(1-pow(y/200/RAD, 2)/2) ) ]=200;
            terrain_temp[ (int) ((xS-xT)/2+((range+1_scan)/2+m)*y/200/RAD) ]
                [ (int) ((yS-yT)/2+((range+1_scan)/2+m)*(1-pow(y/200/RAD, 2)/2) ) ]=200;
        }
        for (y=0; y<(int) (1_scan+6); y++) {
            terrain_temp[ (int) ((xS-xT)/2-m-(range/2+y/2)*phi/2/RAD) ]
                [ (int) ((yS-yT)/2+(range/2+y/2)*(1-pow(phi/2/RAD, 2)/2) ) ]=200;
            terrain_temp[ (int) ((xS-xT)/2+m+(range/2+y/2)*phi/2/RAD) ]
                [ (int) ((yS-yT)/2+(range/2+y/2)*(1-pow(phi/2/RAD, 2)/2) ) ]=200;
        }
    }
    outip = icopen("area.sji", "w", RIE(1), 2, 625, 625, 1);
    for(y=0; y<ysize; y++) {
        for(m=0; m<xsize; m++) rowbuf[m]=terrain_temp[m][y];
        putrow(outip, rowbuf, y, 0);
    } /* end for */
    iclose(outip);

    /* Read the elevation data file */
    if((e_file = fopen(elevation_file, "r")) == NULL ) {
        fprintf(stderr, "Can't open elevation file %s\n", elevation_file);
        exit(1);
    }
    if (fread(charbuf, 1, 12, e_file)==12) {
        format=(long int *) (&charbuf[0]);
        /* density=(float *) (&charbuf[4]);
        size=(long int *) (&charbuf[8]); */
    } /* end if */
    max=-MAX_INT;

```

```

min=MAX_INT;

for(y=0;y<ysize;y++){
  for (m=0;m<xsize;m++){
    if (fread(charbuf,format,1,e_file)!=NULL) {
      elevation[m][y]=*(float *)&charbuf[0];
      if (elevation[m][y]>max) max=elevation[m][y];
      if (elevation[m][y]<min) min=elevation[m][y];
    } /* end if */
    else fprintf(stderr,"Can't read elevation file %s\n",elevation_file);
  } /* end for */
} /* end for */
fclose(t_file);
fclose(image);
fclose(e_file);
printf("\nMinimum elevation of the map=%f m\n",min);
printf("Maximum elevation of the map=%f m\n",max);
for(y=0;y<ysize;y++)
  for (m=0;m<xsize;m++){
    elevation[m][y]=elevation[m][y]-min;
  }
zS=zS-min;
/*display the elevation map*/
/* preposition(2,1+xsize,2,1+ysize);

winopen("Elevations");
RGBmode();
gconfig();
c3s(blackvec);
clear();
for (y=0;y<ysize;y++)
  for (m=0; m<xsize; m++) {
    auxi=(elevation[m][y]-min)*255/(max-min);
    if (auxi<0) auxi=0;
    if (auxi>255) auxi=255;
    draw_a_point(m,y,auxi);
  }
*/
} /* end load_terrain_map */

/*****
float get_sigma(type, angle)
*****/
int type;
float angle;
{
  int i = 0;
  float angl, ang2, sig1, sig2, sigma;

/* get sigma for given angle from typefile */

  angl = data[type][i][0];
  sig1 = data[type][i][1];
  i++;

  while(i <= num[type]){
    ang2 = data[type][i][0];
    sig2 = data[type][i][1];

    if((angle >= angl) && (angle <= ang2)){
      sigma = ((sig2 - sig1) * ((angle - angl)/(ang2 - angl))) + sig1;
      sigma = pow(10, (sigma/10.0));
      return(sigma);
    }
  }
}

```

```

        } /* end if */
    else {
        angl = ang2;
        sig1 = sig2;
        i++;
    } /* end else */
} /* end while */

/* in case there was no angle... */
fprintf(stderr, "No matching angle found in sigma search. %f\t%d\n"
, angle,type);
exit(1);
} /* end get_sigma */

/*****
void read_input_data(input_name)
*****/
char input_name[];
{
    FILE *input;

    if ((input = fopen(input_name, "r")) == NULL) {
        fprintf(stderr, "Can't open %s\n", input_name);
        exit(1);
    }
    printf("\nRead the input data\n");
    fscanf(input, "%f %f %d %f\n", &freq, &dfreq, &n_total_freq,
        &sensitivity);
    fscanf(input, "%f %f %d\n", &zS, &range, &over_sampling);
    fscanf(input, "%f %f %f %f\n", &l_scan, &phi, &az_step_size, &beam_width);
    fscanf(input, "%s %s %d %d %f %f\n", terrain_file, elevation_file, &xsize, &ysize,
        &xyscale, &zscale);
    fscanf(input, "%d %d %f %d %d\n", &xS, &yS, &angle, &xT, &yT);
    fscanf(input, "%s %d %d", outfile_name, &xmagnification,
        &ymagnification);
    fclose(input);
} /* end read_input_data */

/*****
void open_summary_file()
*****/
{
    res = fopen("profile.res", "w");
    fprintf(res, "\n*****\n");
    fprintf(res, "\n          Real sensor image synthesis\n");
    fprintf(res, "\n*****\n");
    fprintf(res, "Sensor height = %.1f m\n", zS);
    fprintf(res, "Range length of area to be scanned = %.1f m\n", l_scan);
    fprintf(res, "Sensor distance to bottom left start of scan = %.1f m\n", range);
    fprintf(res, "Reference frequency = %.1f GHz\n", freq);
    fprintf(res, "Frequency step = %.1f Mhz\n", dfreq);
    fprintf(res, "Number of frequency steps = %d\n", n_total_freq);
    fprintf(res, "Sensitivity = %f dB\n", -10*sensitivity);
    fprintf(res, "Distance per range step = %.2f m\n", d_step_size);
    fprintf(res, "Oversampling factor = %d times\n", over_sampling);
    fprintf(res, "Azimuthal length of scan = %.2f degrees\n", phi);
    fprintf(res, "Azimuthal step size = %.2f degrees\n", az_step_size);
    fprintf(res, "Beam width of sensor = %.2f degrees\n", beam_width);
    fprintf(res, "Terrain map filename = %s\n", terrain_file);
    fprintf(res, "Terrain elevation data filename = %s\n", elevation_file);
    fprintf(res, "Terrain file width = %d pixels\n", xsize);
    fprintf(res, "Terrain file height = %d pixels\n", ysize);
}

```

```

fprintf(res,"Horizontal Scale = %.1f m/pixel\n",xyscale);
fprintf(res,"Vertical Scale = %.1f m/pixel\n",zscale);
fprintf(res,"Sensor location [global x,y] = %d , %d meters\n",xS,yS);
fprintf(res,"Sensor angle from north = 0 : %.2f degrees\n",angle);
fprintf(res,"Global x,y location of terrain map origin = %d , %d meters\n"
        ,xT,yT);
fprintf(res,"Output sgi File = %s\n",outfilename);
fprintf(res,"X magnification of output = %d times\n",xmagnification);
fprintf(res,"Y magnification of output = %d times\n",ymagnification);
fprintf(res,"Output image size: %d X %d pixels\n",
        (scanned width)*xmagnification,(out height)*ymagnification*n total freq);
fprintf(res,"*****\n\n");
fprintf(res,"Terrain number      Filename\n");

}/* end open_summary_file */

/*****/
void initializations()
/*****/
{
    int x,y;
    float d_far;

    for (x=0;x<MAX_ARRAY;x++){
        for (y=0;y<MAX_ARRAY;y++){
            terrain[x][y]=0;
        }/* end for y */
    }/* end for x */
    for (x=0;x<MAX_ARRAY;x++){
        start_index[x]=0;
        for (y=0;y<MAX_ARRAY;y++){
            elevation[x][y]=0;
            ter_cell[x][y].ter=9;
            ter_cell[x][y].hght=0;
            ter_cell[x][y].shadow=0;
            ter_cell[x][y].slope=0;
        }
    }
    /* calculate dimensions */
    d_near = range; /* in meters */
    d_far = d_near + l_scan;
    d_mean = (d_near + d_far) / 2;

    d_step_size=3*pow(10,2)/2/dfreq/n_total_freq*cos(atan(zS/d_mean));

    /* calculate output file size and allocate memory for it */
    out_width = phi/az_step_size;
    out_height = l_scan/d_step_size;
    scanned_width = out_width-beam_width/az_step_size+1;

    dbuf = (float *)malloc((out_width+1) * sizeof(float)); /*dbuf is one line of data,
    horizontally*/
    if(dbuf == NULL){
        printf("Out of memory allocating dbuf\n");
        exit(1);
    }
    dptr = dbuf;
}/* end initializations */

```

```

/*****
void fill_sigma_arrays()
*****/
{
    FILE *namefile,*typefile;
    int index,m;
    float angtmp,sigtmp;
    char name[12];

/* get name of file containing type info */
    if ((namefile = fopen("filename.dat", "r")) == NULL ){
        fprintf(stderr, "Can't open filename.dat\n");
        exit(1);
    }/* end if */
    printf("\nGenerate look-up tables\n");
    index = 0;
    while(fscanf(namefile,"%d %s", &index, name) != EOF){
        if ((typefile = fopen(name, "r")) == NULL ){
            fprintf(stderr, "Can't open typefile\n");
            exit(1);
        }/* end if */
        fprintf(res, "      %d          %s\n", index,name);
        m = 0;
        while (fscanf(typefile, "%f %f", &angtmp, &sigtmp) != EOF){
            data[index][m][0] = angtmp;
            data[index][m][1] = sigtmp;
            m++;
        }
        num[index] = m - 1;
        fclose(typefile);
        index++;
    }
    fclose(namefile);
}/* end fill_sigma_arrays */

/*****
void first_scanning_process()
*****/
{
    float slope_min,slope_max;
    int x,y,y0;
    float beam_angle,sin_beam,cos_beam;
    float next_min_height,dist;
    int enough;
    float depression,shadow_height_step;
    struct point_coord inter[4],intersection[2],radar,aux,current_pt;
    struct corner_struct corner[4];
    float dx,dy,slope,elev,distance;
    float dis[4],xP, yP, zP,alpha,beta,gamma,xw, yw,x1,x2,y1,y2,z1,z2;
    int indice[2];
    int x_coord, y_coord;
    int number,loop,index_min,min_dist,area_reached;

    printf("\nPre-processing of the input data\n");
    slope_min=180;
    slope_max=-180;
    hghtmax=MAX_INT;
    hghtmin=MAX_INT;
    y_max=0;
    x_min=MAX_INT;
    y_min=MAX_INT;
    y_max=-MAX_INT;
    x_max=-MAX_INT;
}

```

```

radar.x = xS;
radar.y = yS;
radar.z = zS;

/* First scanning process : profile lines */

for (x=0; x<out_width; x++) {

    beam_angle = angle-phi/2 + (az_step_size * x);
    sin_beam = fsin(beam_angle/RAD);
    cos_beam = fcos(beam_angle/RAD);
    enough=0;
    next_min_height=0;
    area_reached=0;

    for (y=0; enough==0; y++){
        /* For each distance step for length of scan */
        /* Start at nearest point, move to farthest */

        if (area_reached) dist = d_near + (d_step_size/over_sampling * (float) (y));
        else {
            if (cos_beam!=0) dist = abs((yT-
yS)/cos_beam)+(float) (y)*d_step_size/over_sampling;
            else dist = abs(xT-xS)+(float) (y)*d_step_size/over_sampling;
        }

        do
        {
            /* when the point is exactly on a terrain grid point */
            /* the slope cannot be computed */

            /* get point location in world coords */
            xP = xS + sin_beam * dist;
            yP = yS + cos_beam * dist;

            /* convert to integer terrain map coords */
            x_coord = (xP - xT)/xyscale;
            y_coord = (yP - yT)/xyscale;

            if (x_coord<x_cmin) x_cmin=x_coord;
            if (x_coord>x_cmax) x_cmax=x_coord;
            if (y_coord<y_cmin) y_cmin=y_coord;
            if (y_coord>y_cmax) y_cmax=y_coord;

            /* convert back to world coords left bottom corner */
            xw = x_coord * xyscale + xT;
            yw = y_coord * xyscale + yT;

            /* we move the point a little bit */
            if ((xw==xP) && (yw==yP))
                dist=dist-d_step_size/over_sampling/10;
        } while ((xw==xP) && (yw==yP));

        /* We scan out of the map */
        if ((x_coord<0) || (y_coord<0) || (x_coord>=xsize) ||
(y_coord>=ysize)) {
            fprintf(stderr, "Scanning out of the map\n");
            exit(1);
        } /* end if */

        /* We scan out of the maximum size of the array */
        if ((x_coord>=MAX_ARRAY) || (y_coord>=MAX_ARRAY)
|| (x>MAX_ARRAYX) || (y>MAX_ARRAYY)) {
            fprintf(stderr, "Out of memory scan\n");

```

```

        exit(1);
    }/* end if */

/* beam plan equation : alpha * x + beta * y = gamma */
alpha = cos_beam;
beta = -sin_beam;
gamma = xS*cos_beam-yS*sin_beam;

current_pt.x = xP;
current_pt.y = yP;
comer[0].x = xw;
comer[0].y = yw+xyscale;
comer[0].ter = terrain[x_coord][y_coord+1];
comer[1].x = xw+xyscale;
comer[1].y = yw+xyscale;
comer[1].ter = terrain[x_coord+1][y_coord+1];
comer[2].x = xw+xyscale;
comer[2].y = yw;
comer[2].ter = terrain[x_coord+1][y_coord];
comer[3].x = xw;
comer[3].y = yw;
comer[3].ter = terrain[x_coord][y_coord];

/* calculate the intersection points of this plan and */
/* the terrain cell rectangle */

/* calculate the word coord of the four possible */
/* intersection points */
if (alpha!=0) {
    /* when the beam is not parallel to xz */
    x1=xw+xyscale;
    x2=xw;
    y1=yw;
    inter[1].x = (gamma - beta * y1)/alpha;
    inter[1].y = y1;
    z1=elevation[x_coord+1][y_coord]*zscale;
    z2=elevation[x_coord][y_coord]*zscale;
    inter[1].z = z2+(inter[1].x-x2)*(z1-z2)/(x1-x2);

    x1=xw;
    x2=xw+xyscale;
    y1=yw+xyscale;
    inter[3].x = (gamma - beta * y1)/alpha;
    inter[3].y = y1;
    z1=elevation[x_coord][y_coord+1]*zscale;
    z2=elevation[x_coord+1][y_coord+1]*zscale;
    inter[3].z = z2+(inter[3].x-x2)*(z1-z2)/(x1-x2);
}/* end if */
if (beta!=0) {
    /* when the beam is not parallel to yz */
    y1=yw;
    y2=yw+xyscale;
    x1=xw;
    inter[0].x = x1;
    inter[0].y = (gamma - alpha * x1)/beta;
    z1=elevation[x_coord][y_coord]*zscale;
    z2=elevation[x_coord][y_coord+1]*zscale;
    inter[0].z = z2+(inter[0].y-y2)*(z1-z2)/(y1-y2);

    y1=yw+xyscale;
    y2=yw;
    x1=xw+xyscale;
    inter[2].x = x1;
    inter[2].y = (gamma - alpha * x1)/beta;

```



```

z1=elevation[x_coord+1][y_coord+1]*zscale;
z2=elevation[x_coord+1][y_coord]*zscale;
inter[2].z = z2+(inter[2].y-y2)*(z1-z2)/(y1-y2);
}/* end if */

if ((alpha!=0) && (beta!=0)) {
/* determine the only two real intersection points */
/* when the beam is parallel neither to xz nor to yz */
x1=xw;
x2=xw+xyscale;
y1=yw;
y2=yw+xyscale;
number=0;
for (loop=0;loop<4;loop++){
if ((inter[loop].x<=x2) && (inter[loop].x>=x1) &&
(inter[loop].y<=y2) && (inter[loop].y>=y1)) {
indice[number]=loop;
number++;
}/* end if */
}/* end for */
for (loop=0; loop<number;loop++)
intersection[loop]=inter[indice[loop]];
}/* end if */
else if (alpha==0) {
/* when the beam is parallel to xz */
intersection[0]=inter[0];
intersection[1]=inter[2];
}/* end if */
else if (beta==0) {
/* when the beam is parallel to yz */
intersection[0]=inter[1];
intersection[1]=inter[3];
}/* end if */

dis[0]=compute_dist(intersection[0],radar);
dis[1]=compute_dist(intersection[1],radar);
if (dis[0] > dis[1]) {
aux=intersection[0];
intersection[0]=intersection[1];
intersection[1]=aux;
}/* end if */

/* determine the corresponding height and slope */
elev=intersection[1].z-intersection[0].z;
distance=compute_dist(intersection[0],intersection[1]);
if (distance==0) {
zP=intersection[0].z;
slope=0;
}/*end if*/
else {
zP=intersection[0].z+elev*
compute_dist(current_pt,intersection[0])/distance;
slope=atan(elev/distance)*RAD;
}/*end else*/
current_pt.z=zP;

if (area_reached==0) {
shadow_height_step=(d_near-dist)*(zS-zP)/dist;

if (next_min_height<zP-shadow_height_step)
next_min_height = zP-shadow_height_step;
if ((d_near-dist)<d_step_size/over_sampling) {
area_reached=1;
shadow_height_step=d_step_size/over_sampling*(zS-next_min_height)/d_near;
}
}
}

```

```

        y=-1;
        /* end if */
    }/* end if */

if (area_reached) {
    /* determine the corresponding terrain type */
    dis[0]=compute_dist(corner[0],current_pt);
    dis[1]=compute_dist(corner[1],current_pt);
    dis[2]=compute_dist(corner[2],current_pt);
    dis[3]=compute_dist(corner[3],current_pt);
    min_dist=xyscale;
    for (loop=0;loop<=3;loop++) {
        if (dis[loop]<min_dist) {
            min_dist = dis[loop];
            index_min = loop;
        }/* end if */
    }/* end for */
    /* determine the type of shadowing of this terrain cell */
    dist=compute_dist(current_pt,radar);
    depression=atan((zS-zP)/dist)*RAD;
    if (zP<next_min_height) {
        ter_cell[x][y].shadow=2;
        next_min_height=next_min_height-shadow_height_step;
    }/* end if */
    else {
        ter_cell[x][y].shadow=0;
        shadow_height_step=d_step_size/over_sampling*(zS-zP)/dist;
        next_min_height=zP-shadow_height_step;
    }/* end else */
    if ((depression+slope <= 0) || ((slope<0) &&
        (ter_cell[x][y].shadow==2))) ter_cell[x][y].shadow=1;

    /* fill the array with the values found */
    ter_cell[x][y].slope=slope;
    ter_cell[x][y].hght=zP;
    ter_cell[x][y].ter=comer[index_min].ter;

    /* Tree lines */
    if ((ter_cell[x][y].ter==4) && (ter_cell[x][y-1].ter!=4)
        && (ter_cell[x][y-1].ter!=5)) {
        y0=y;
        do {
            ter_cell[x][y0].ter=5;
            y0=y0-1;
        }
        while ((y-y0)*(d_step_size/over_sampling)<TREE_LINE_THICKNESS);
    }/* end if */

    /* Maximum and minimum */
    if (slope< slope_min) slope_min=slope;
    if (slope> slope_max) slope_max=slope;
    if (zP>hghtmax) hghtmax=zP;
    if (zP<hghtmin) hghtmin=zP;

    /* End of scan */
    if (zP<(y/over_sampling-out_height-1)*d_step_size*d_mean/zS) {
        enough=1;
        if (y>y_max) y_max=y;
    }/* end if */
    }/* end if (area_reached) */
} /* end for y */
} /* end for x */

```

```
printf("\nFor the scanned region:\n");
```

```

printf("Minimum elevation=%f m\n",hghtmin+min);
printf("Maximum elevation=%f m\n",hghtmax+min);
printf("Minimum slope=%f deg\n",slope_min);
printf("Maximum slope=%f deg\n",slope_max);

for (y=0;y<y_max;y++)
  for (x=0;x<out_width;x++)
    ter_cell[x][y].hght=ter_cell[x][y].hght-hghtmin;

zS=zS-hghtmin;
}/* end first_scaming_process */

/*****/
void display_windows()
/*****/
{
  int x,y,auxi;

  prefposition(1,out_width,600,y_max/over_sampling+600);
  winopen("Shadows");
  RGBnode();
  gconfig();
  c3s(blackvec);
  clear();
  for (y=0;y<y_max/over_sampling;y++)
    for (x=0;x<out_width;x++) {
      if (ter_cell[x][y*over_sampling].shadow!=0) {
        auxi=ter_cell[x][y*over_sampling].shadw*100;
        draw_a_point(x,y,auxi);
      }/* end if */
    }/* end for */

  prefposition(22+out_width,21+2*out_width,y_max/over_sampling+600,
              2*y_max/over_sampling+600);
  winopen("Elevations");
  RGBnode();
  gconfig();
  c3s(blackvec);
  clear();
  for (y=0;y<y_max/over_sampling;y++)
    for (x=0;x<out_width;x++) {
      auxi=(ter_cell[x][y*over_sampling].hght)*255/(hghtmax-hghtmin);
      draw_a_point(x,y,auxi);
    }/* end for */

  prefposition(38+2*out_width,37+3*out_width,600,y_cmax-y_cmin+600);
  winopen("True Elevations");
  RGBnode();
  gconfig();
  c3s(blackvec);
  clear();
  for (y=0;y<y_cmax-y_cmin;y++)
    for (x=0;x<x_cmax-x_cmin;x++) {
      auxi=(elevation[x+x_cmin][y+y_cmin]-hghtmin)*255/(hghtmax-hghtmin);
      if (auxi<0) auxi=0;
      if (auxi>255) auxi=255;
      draw_a_point(x,y,auxi);
    }/* end for */

  prefposition(54+3*out_width,53+4*out_width,600,y_max/over_sampling+600);
  winopen("Terrain");
  RGBnode();

```

```

gconfig();
c3s(blackvec);
clear();
for (y=0;y<y_max/over_sampling;y++)
  for (x=0; x<out_width; x++) {
    auxi=(ter_cell[x][y*over_sampling].ter-0)*255/(9-0);
    if (auxi<0) auxi=0;
    if (auxi>255) auxi=255;
    draw_a_point(x,y,auxi);
  }/* end for */

}/* end display_windows */

/*****
void draw_a_point(c1,c2,niv)
*****/
int c1,c2,niv;
{ long int vert[2];

  rgbvector[0]=niv;
  rgbvector[1]=rgbvector[0];
  rgbvector[2]=rgbvector[0];
  c3s(rgbvector);
  bgrpoint();
  vert[0]=c1;
  vert[1]=c2;
  v2i(vert);
  endpoint();

}/* end draw_a_point */

/*****
void set_weight_array()
*****/
{
  int seed=-1,x,y;

  for(x=0;x<out_width;x++){
    for(y=0;y<y_max;y++){
      weight_array[x][y][0] = set_random_array(&seed);
      weight_array[x][y][1] = set_random_array(&seed);
    }/* end for */
  }/* end for */

}/* end set_weight_array */

/*****
void coherent_summation()
*****/
{
  FILE *output;
  float k, weight_a, weight_b, sigma,wl;n;
  fcomplex signal, retsig, weight, point_signal;
  int x,y,samplex,sampley,n;
  float rad, shadow,step_sum;
  double point_pow;
  float distance;
  int flag_end;
  float elev_min,elev_max,area;
  int bw_samples;
  float new_angle, az_angle;

```

```

maximum=MAX_INT;
minimum=MAX_INT;

/* scattered field = coherent sum of returns */

printf("\nCoherent summation of the return and computation of the power\n");

output = fopen("profile.out", "w");

/* determine k */
wvln = 0.3/freq; /* wavelength = speed of light/frequency */
k = (2*PI)/wvln;

for (y=0; y < out_height; y++) {
/* For each distance step for length of scan */
dist = d_near + (d_step_size * y);
/* Start at nearest point, move to farthest */
dptr = dbuf;

for (x=0; x<scanned_width; x++){

step_sum = 0.0;
retsig.r = 0.0;
retsig.i = 0.0;
az_angle = (angle - phi/2) + (az_step_size * x);
bw_samples = beam_width/az_step_size;

for (samplex=0;samplex<bw_samples;samplex++) {
flag_end=0;
for (sampley=start_index[x+samplex];flag_end==0;sampley++) {
point_signal.r=0;
point_signal.i=0;
if (sampley==start_index[x+samplex]) {
do {
distance=(float) (sampley)/over_sampling*d_step_size-y*d_step_size;
elev_max=distance*d_mean/zS;
sampley++;
} while (ter_cell[x+samplex][sampley-1].hght>elev_max);
sampley--;
start_index[x+samplex]=sampley;
}/* end if */

distance=(float) (sampley)/over_sampling*d_step_size-y*d_step_size;
elev_max=distance*(d_mean/zS);
elev_min=(distance-d_step_size) * (d_mean/zS);

rad=(distance+dist)/cos(atan(zS/ (dist+d_step_size/2)))
-ter_cell[x+samplex][sampley].hght/sin(atan(zS/ (dist+d_step_size/2))));

signal = RCmul(d_near/rad,Complex(cos(2*k*rad), sin(2*k*rad)));
if ((ter_cell[x+samplex][sampley].hght<=elev_max) &&
(ter_cell[x+samplex][sampley].hght>elev_min))
{
if (ter_cell[x+samplex][sampley].shadow!=1) {
new_angle = fatan((distance+dist)/
(zS-ter_cell[x+samplex][sampley].hght)) *RAD;
sigma = get_sigma(ter_cell[x+samplex][sampley].ter,
new_angle-ter_cell[x+samplex][sampley].slope);
/* get weight */
weight_a = weight_array[x+samplex][sampley][0];
weight_b = weight_array[x+samplex][sampley][1];
weight = RCmul(1/sqrt(2),RCmul(sqrt(sigma),
Complex(weight_a, weight_b)));
}
}
}
}
}
}

```

```

        area=d_step_size/over_sampling*
            cos(ter_cell[x+samplex][sampley].slope/RAD)*
            az_step_size/RAD*(dist+distance)*pow(xyscale,2);
        point_signal=RCmul(RCmul(sqrt(area),weight), signal);
        if (ter_cell[x+samplex][sampley].shadow==2)
            point_signal=RCmul(ATT_SHAD2,point_signal);
        /* end if */
    else point_signal=Complex(ATT_SHAD1,0);
    /* end if */
else flag_end=1;

    retsig = Cadd(retsig, point_signal);
    /*end for sampley*/
/* end for samplex */

point_pow = (pow(retsig.r,2) + pow(retsig.i,2));
step_sum = 10*flog10(point_pow);
if (step_sum<-10*sensitivity) step_sum=-10*sensitivity;

if ((step_sum < minimum) && (step_sum!==-10000)) minimum = step_sum;
if(step_sum > maximum) maximum = step_sum;

*dptr = step_sum;
dptr++;
} /* end for x */

dptr = dbuf;
for(n=0;n<scanned_width;n++){
    fprintf(output, "%f\n", *dptr);
    dptr++;
}
}/* end for y */
fclose(output);

}/* end coherent_summation */

/*****/
void create_output ()
/*****/
{
    float sat1;
    float sat2;
    FILE *histname;
    FILE *inprofile;
    float bin_step;
    int x,y,n;
    int xtimes,ytimes;
    unsigned short *outbuf, *outptr, outvalue;
    float value;
    IMAGE *outip;
    float fmin, nmin;
    int sum;
    int bin_val[255];
    /* char cbuf[4096]; for binary file */

    printf("\nGenerate the synthesized image\n");

    fprintf(res, "\nminimum output = %f dB\nmaximum output = %f dB\n\n",minimum,maximum);

/* now read profile.out in and use to output an sgi format image file */

/*outip = fopen(outfilename, "w"); to use for binary format*/

```

```

outip = fopen(outfilename, "w", RLE(1), 2, scanned_width*xmagnification,
              out_height*y_magnification, 1);

outbuf = (unsigned short *)malloc(scanned_width * xmagnification *
                                  sizeof(unsigned short));
if(outbuf == NULL) {
    fprintf(stderr, "Could not alloc outbuf!\n");
    exit(-1);
}

if ((inprofile = fopen("profile.out", "r")) == NULL) {
    fprintf(stderr, "Ideal: can't open profile.out\n");
    exit(1);
}

bin_step = (maximum-minimum)/255;

for (n=0;n<255;n++) bin_val[n]=0;

for(y=0;y<out_height;y++) {

    for(x=0;x<scanned_width;x++) {
        fscanf(inprofile,"%f\n", &value);
        for(n=0;n<255;n++) {
            if((value>=(minimum+n*bin_step)) &&
                (value<(minimum+(n+1)*bin_step))) {
                bin_val[n]=bin_val[n]+1;
            } /* end if */
        } /* end for n */
        if (value>=maximum) bin_val[254]++;
    } /* end for x */
} /* end for y */

/* for histogram */
sum=0;

sat1=minimum+bin_step;
sat2=maximum-bin_step;
histname=fopen("hist.dat","w");
for(n=0;n<255;n++) {
    fmin=minimum+n*bin_step;
    nmin=minimum+(n+1)*bin_step;
    fprintf(histname,"%f<v<%f\t%d\n", fmin, nmin, bin_val[n]);
    sum=sum+bin_val[n];
    if (sum<(0.015*out_width*out_height+bin_val[0])) sat1=fmin+bin_step;
    if (sum<(0.99*out_width*out_height)) sat2=nmin+bin_step;
} /* end for n */
fclose(histname);
fprintf(res, "Lower saturation value = %.1f dB\n", sat1);
fprintf(res, "Upper saturation value = %.1f dB\n", sat2);

rewind(inprofile);
for(y=0;y<out_height;y++) {
    outptr = outbuf;
    for(x=0;x<scanned_width;x++) {
        fscanf(inprofile,"%f\n", &value);
        if(value<sat1){
            outvalue = 0;
        } /* end if */
        else if(value>sat2) outvalue = 255;
        else{
            outvalue = (255/(sat2-sat1))*(value-sat1);
        } /* end else */
        for(xtimes = 0; xtimes<xmagnification;xtimes++, outptr++)

```

```

        *outptr = outvalue;
    }/* end for x */
    for(ytimes = 0;ytimes<ymagnification;ytimes++){

        /*stoc(outbuf,dhbuf,out_width*xmagnification);
        fwrite(dhbuf,1,out_width*xmagnification,outip);
        to use for binary format */

        putrow(outip, outbuf, y*xmagnification + ytimes, 0);
    } /* end for ytimes */
}/* end for y */

fclose(outip);
fclose(inprofile);
fclose(res);
free(dbuf);
free(outbuf);
}

```


Appendix C

C CODE MODIFICATIONS FOR REAL SENSOR IMAGE SYNTHESIS

The program corresponding to the real sensor simulation algorithm only differ from the idealized synthesis program in three functions:

- initializations.
- coherent_summation.
- create_output.

The following code lists the changes.

```
/******  
void initializations()  
/******  
{  
  int x,y;  
  float d_far;  
  
  for (x=0;x<MAX_ARRAY;x++){  
    for (y=0;y<MAX_ARRAY;y++){  
      terrain[x][y]=0;  
    }/* end for y */  
  }/* end for x */  
  for (x=0;x<MAX_ARRAYX;x++){  
    start_index[x]=0;  
    for (y=0;y<MAX_ARRAYY;y++){  
      elevation[x][y]=0;  
      ter_cell[x][y].ter=9;  
      ter_cell[x][y].hght=0;  
      ter_cell[x][y].shadow=0;  
      ter_cell[x][y].slope=0;  
    }  
  }  
}
```

```

/* calculate dimensions */
d_near = range;          /* in meters */
d_far = d_near + l_scan;
d_mean = (d_near + d_far) / 2;

d_step_size=3*pow(10,2)/2/dfreq*cos(atan(zS/d_mean));

/* calculate output file size and allocate memory for it */
out_width = phi/az_step_size;
out_height = l_scan/d_step_size;
scanned_width = out_width-beam_width/az_step_size+1;

dbuf = (float *)malloc((out_width+1) * sizeof(float));/*dbuf is one line of data,
horizontally*/
if(dbuf == NULL){
    printf("Out of memory allocating dbuf\n");
    free(dbuf);
    exit(1);
}
dptr = dbuf;

}/* end initializations */

/*****
void coherent_summation()
*****/
{
    FILE *output;
    float k, weight_a, weight_b, w, sigma;
    fcomplex signal, weight, point_signal, field_tmp;
    int x,y,samplex,sampley,n;
    float rad, shadbw;
    double point_pow;
    float distance;
    int flag_end;
    float elev_min,elev_max,area;
    int bw_samples;
    float new_angle, az_angle;
    float targ_pow;
    int nfreq;
    float rtm[1024];
    fcomplex rtm2[1024],rtm3[1024];
    float step_sum[MAX_ARRAY][MAX_ARRAY],r,f_tmp;

    maximum=MAX_INT;
    minimum=MAX_INT;

/* scattered field = coherent sum of returns */

    printf("\nCoherent summation of the return and computation of the power\n");

    output = fopen("profile.out", "w");

    for (y=0; y < out_height; y++){
        /* For each distance step for length of scan */
        dist = d_near + (d_step_size * y);
        /* Start at nearest point, move to farthest */
        dptr = dbuf;

        for (x=0; x<scanned_width; x++){

            for (nfreq=0;nfreq<n_total_freq;nfreq++){
                rtm[2*nfreq+1]= 0;
            }
        }
    }
}

```

```

        rtm[2*nfreq+2]= 0;
    }

    az_angle = (angle - phi/2) + (az_step_size * x);
    bw_samples = beam_width/az_step_size;

    for (samplex=0;samplex<bw_samples;samplex++) {
        flag_end=0;
        for (sampley=start_index[x+samplex];flag_end==0;sampley++) {
            point_signal.r=0;
            point_signal.i=0;
            if (sampley==start_index[x+samplex]) {
                do {
                    distance=(float) (sampley)/over_sampling*d_step_size-y*d_step_size;
                    elev_max=distance*(d_mean/zS);
                    sampley++;
                } while (ter_cell[x+samplex][sampley-1].hght>elev_max);
                sampley--;
                start_index[x+samplex]=sampley;
            }/* end if */

            distance=(float) (sampley)/over_sampling*d_step_size-y*d_step_size;
            elev_max=distance*(d_mean/zS);
            elev_min=(distance-d_step_size)*(d_mean/zS);
            rad=(distance+dist)/cos(atan(zS/(dist+d_step_size/2)))
            -ter_cell[x+samplex][sampley].hght/sin(atan(zS/(dist+d_step_size/2)));

            if ((ter_cell[x+samplex][sampley].hght<=elev_max) &&
                (ter_cell[x+samplex][sampley].hght>elev_min))
            {
                if (ter_cell[x+samplex][sampley].shadow!=1) {
                    new_angle = fatan((distance+dist)/
                        (zS-ter_cell[x+samplex][sampley].hght))*RAD;
                    sigma = get_sigma(ter_cell[x+samplex][sampley].ter,
                        new_angle-ter_cell[x+samplex][sampley].slope);
                    /* get weight */
                    weight_a = weight_array[x+samplex][sampley][0];
                    weight_b = weight_array[x+samplex][sampley][1];
                    weight = RCmul(1/sqrt(2),RCmul(sqrt(sigma),
                        Complex(weight_a, weight_b)));
                    area=d_step_size/over_sampling*
                        cos(ter_cell[x+samplex][sampley].slope/RAD)*
                        az_step_size/RAD*(dist+distance)*pow(xyscale,2);
                    field_tmp=RCmul(sqrt(area),weight);
                    for (nfreq=0;nfreq<n_total_freq;nfreq++){
                        /* determine k */
                        k = (2*PI)*(1000*freq+nfreq*dfreq)/300;
                        signal = Complex(cos(2*k*rad),sin(2*k*rad));
                        point_signal = Cmul(field_tmp,signal);
                        if (ter_cell[x+samplex][sampley].shadow==2)
                            point_signal=RCmul(ATT_SHAD2,point_signal);
                        rtm[2*nfreq+1]= rtm[2*nfreq+1]+point_signal.r;
                        rtm[2*nfreq+2]= rtm[2*nfreq+2]+point_signal.i;
                    }/*end nfreq*/
                }/* end if */
            }
            else {
                for (nfreq=0;nfreq<n_total_freq;nfreq++){
                    point_signal=Complex(ATT_SHAD1,0);
                    rtm[2*nfreq+1]= rtm[2*nfreq+1]+point_signal.r;
                    rtm[2*nfreq+2]= rtm[2*nfreq+2]+point_signal.i;
                }/*end nfreq */
            }
            }/* end if */
        }
    }
}

```

```

        flag_end=1;
        /*      start_index[x+samplex]=sampley-1;*/
    }
} /*end for samplex*/
}/* end for samplex */

r=dist/cos(atan(zS/(dist+d_step_size/2)));
for (n=0;n<n_total_freq;n++){
    rtm2[n]=Complex(rtm[2*n+1],rtm[2*n+2]);
    rtm2[n]=Cmul(rtm2[n],Complex(cos(2*n*2*PI*dfreq/300*r),
        -sin(2*n*2*PI*dfreq/300*r)));
}/* end for */
for (n=0;n<n_total_freq;n++){
    rtm3[n].r=0;
    rtm3[n].i=0;
    for (nfreq=0;nfreq<n_total_freq;nfreq++){
        rtm3[n]=Cadd(rtm3[n],Cmul(rtm2[nfreq],
            Complex(cos(2*PI/n_total_freq*nfreq*n),
                -sin(2*PI/n_total_freq*nfreq*n))));
        rtm[2*n+1]=rtm3[n].r;
        rtm[2*n+2]=rtm3[n].i;
    }/* end for*/

for (nfreq=0;nfreq<n_total_freq;nfreq++){
    point_pow = (pow(rtm[2*nfreq+1],2) + pow(rtm[2*nfreq+2],2));

    step_sum [x] [nfreq]= 10*flog10(point_pow)-20*flog10(n_total_freq);

    if (step_sum[x] [nfreq]<-10*sensitivity)
        step_sum[x] [nfreq]=-10*sensitivity;
    if (step_sum[x] [nfreq] < minimum) minimum = step_sum[x] [nfreq];
    if(step_sum[x] [nfreq] > maximum) maximum = step_sum[x] [nfreq];
} /* end for nfreq */
} /* end for x */
for (nfreq=0;nfreq<n_total_freq;nfreq++){
    for(x=0; x<scanned_width; x++){
        fprintf(output, "%f\n",step_sum[x] [nfreq]);
    }/* end for x */
}/* end for nfreq */

}/* end for y */
fclose(output);

}/* end coherent_summation */

/*****/
void create_output ()
/*****/
{
    float sat1;
    float sat2;
    FILE *histname;
    FILE *inprofile;
    float bin_step;
    int x,y,n;
    int xtimes,ytimes;
    unsigned short *outbuf, *outptr, outvalue;
    float value;
    IMAGE *outip;
    float fmin, nmin;
    int sum;
    int bin_val[255];
    /* char chbuf[4096]; for binary file */

```

```

printf("\nGenerate the synthesized image\n");

fprintf(res, "\nminimum output = %f dB\nmaximum output = %f dB\n\n", minimum, maximum);

/* now read profile.out in and use to output an sgi format image */

/*outip = fopen(outfilename, "w"); to use for binary format*/

outip = iopen(outfilename, "w", RLE(1), 2, scanned_width*xmagnification,
             out_height*y magnification*n_total_freq, 1);

outbuf = (unsigned short *)malloc(scanned_width * xmagnification *
                                 sizeof(unsigned short));

if(outbuf == NULL) {
    fprintf(stderr, "Could not alloc outbuf!\n");
    free(outbuf);
    exit(-1);
}

if ((inprofile = fopen("profile.out", "r")) == NULL) {
    fprintf(stderr, "Real: can't open profile.out\n");
    exit(1);
}

bin_step = (maximum-minimum)/255;

for (n=0;n<255;n++) bin_val[n]=0;

for(y=0;y<out_height*n_total_freq;y++) {

    for(x=0;x<scanned_width;x++) {
        fscanf(irprofile, "%f\n", &value);
        for(n=0;n<255;n++) {
            if ((value>=(minimum+n*bin_step)) &&
                (value<(minimum+(n+1)*bin_step))) {
                bin_val[n]=bin_val[n]+1;
            } /* end if */
        } /* end for n */
        if (value>=maximum) bin_val[254]++;
    } /* end for x */
} /* end for y */

/* for histogram */
sum=0;
sat1=minimum+bin_step;
sat2=maximum-bin_step;
histname=fopen("hist.dat", "w");
for(n=0;n<255;n++) {
    fmin=minimum+n*bin_step;
    nmin=minimum+(n+1)*bin_step;
    fprintf(histname, "%f<v<%f\t%d\n", fmin, nmin, bin_val[n]);
    sum=sum+bin_val[n];
    if (sum<(0.015*scanned_width*out_height*n_total_freq+bin_val[0]))
        sat1=fmin+bin_step;
    if (sum<(0.99*scanned_width*out_height*n_total_freq))
        sat2=nmin+bin_step;
} /* end for n */
fclose(histname);
fprintf(res, "Lower saturation value = %.1f dB\n", sat1);
fprintf(res, "Upper saturation value = %.1f dB\n", sat2);

rewind(irprofile);
for(y=0;y<out_height*n_total_freq;y++) {

```

```

outptr = outbuf;
for(x=0;x<scanned_width;x++){
  fscanf(infile,"%f\n", &value);
  if(value<sat1){
    outvalue = 0;
    /* end if */
  } else if(value>sat2) outvalue = 255;
  else{
    outvalue = (255/(sat2-sat1))*(value-sat1);
    /* end else */
  }
  for(xtimes = 0; xtimes<xmagnification;xtimes++, outptr++){
    *outptr = outvalue;
  } /* end for x */
  for(ytimes = 0;ytimes<ymagnification;ytimes++){

    /*stoc(outbuf,dbuf,out_width*xmagnification);
    fwrite(dbuf,1,out_width*xmagnification,outip);
    to use for binary format */

    putrow(outip, outbuf, y*ymagnification + ytimes, 0);
  } /* end for ytimes */
} /* end for y */

fclose(outip);
fclose(infile);
fclose(res);
free(dbuf);
free(outbuf);
}

```

Bibliography

D. K. Barton (1964), "Radar System Analysis," Prentice Hall Inc., Englewood Cliffs, N.J..

D. K. Barton (1988), "Modern Radar System Analysis," Artech House.

D. K. Barton and H. R. Ward (1984), "Handbook of Radar Measurement," Artech House.

M. Borgeaud (Dec. 1987), "Theoretical Models For Polarimetric Microwave Remote Sensing of Earth Terrain," Ph. D. Thesis, Massachusetts Institute of Technology.

N. C. Curie and C. E. Brown (1987), "Principles and Applications of Millimeter-Wave Radar," Artech House.

N. C. Curie, R. D. Hayes, and R. N. Trebits (1992), "Millimeter-Wave Radar Clutter," Artech House.

J. W. Goodman (1979), "Statistical Properties of Laser Speckle Patterns," Laser Speckle and Related Phenomena, Ed. J. C. Dainty, Springer-Verlag.

J. C. Holtzman et al. (Oct. 1978), " Radar Image Simulation," IEEE Trans. Geo. Elec., Vol. GE-16 No. 4, 296-303.

G. K. Huddleston, M. Tanenhaus, and B. P. Williams (1989), "Millimeter-Wave and Synthetic Aperture Radar," Proceedings, SPIE-The International Society for Optical Engineering.

IEEE Standard 521-1976, November 30, 1976.

- A. Kelley, I. Pohl, "A book on C," The Benjamin / Cummings Publishing Company, Inc.
- J. A. Kong (1990), "Electromagnetic Wave Theory," Second Edition, Wiley Interscience.
- J. A. Kong et al. (Sept. 1979), "Theoretical Modeling and Experimental Data Matching for Active and Passive Remote Sensing of Earth Terrain," AGARD Report CPP-269.
- J. A. Kong et al. (1988), "Polarimetric Clutter Modeling: Theory and Application," GACIAC PR-88-03, The Polarimetric Technology Workshop, Rocket Auditorium, Redstone Arsenal, U.S. Army Missile Command, Huntsville, AL, Aug. 16-19.
- D. L. Mensa (1981), "High Resolution Radar Imaging," Artech House, Inc..
- J. F. Reintjes and G. T. Coate (1952), "Principles of Radar," 3d edition, Mc Graw-Hill Book Company, New York.
- A. W. Rihaczek (1969), "Principles of High Resolution Radar," Mc Graw-Hill Book Company.
- S. O. Rice (1963), "Reflection of EM by Slightly Rough Surfaces," The Theory of Electromagnetic Waves, M. Kline, Ed., Interscience, NY.
- R. T. Shin (Aug. 1984), "Theoretical Models For Microwave Remote Sensing of Earth Terrain," Ph. D. Thesis, Massachusetts Institute of Technology.
- M. I. Skolnik (1962), "Introduction to Radar Systems," Mc Graw Hill Book Company Inc., New York.
- M. I. Skolnik (1970), "Radar Handbook," Mc Graw Hill Handbooks, New York.
- A. Stogryn (1974), "EM Scattering by Random Dielectric Constant Fluctuations in Bounded Medium," Radio Science, 9, p. 509-518.

A. A. Swartz et al. (1988), "Radar Range Profile Simulation of Terrain Clutter using the Random Medium Model," The Polarimetric Technology Workshop, Rocket Auditorium, Redstone Arsenal, U.S. Army Missile Command, Huntsville, AL, Aug. 16-19. .

L. Tsang and J. A. Kong (July 1976), "Emissivity of Half Space Random Media," Radio Science, 11, p. 593-598.

L. Tsang, J. A. Kong, and R. T. Shin (1985), "Theory of Microwave Remote Sensing," Wiley Interscience.

G. R. Valenzuela (1968), "Scattering of EM Waves from a Tilted Slightly Rough Surface," Radio Science, 3, p. 1057-1064.

F. J. Vallese (May 1980), "Radar Image Prediction Using Theoretical Terrain-Surface Models," Master Thesis, Massachusetts Institute of Technology.

W. T. Vetterling, S. A. Teukolsky, W. H. Press, B. P. Flannery, "Numerical Recipes Example Book (C)," Cambridge University Press.