NAVAL POSTGRADUATE SCHOOL
Monterey California

THESIS

THREE DIMENSIONAL IMAGE SYNTHESIS:
THEORY AND APPLICATION

by

Charles N. Adams IV

June 2003

Thesis Advisor:	Phillip E. Pace
Co-Advisor:	Don Brutzman

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Three Dimensional Image Synthesis: Theory and Application	**5. FUNDING NUMBERS**
6. AUTHOR Charles N. Adams IV	
7. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Joint Services Electronic Warfare, Naval Postgraduate School, Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME AND ADDRESS Office of Naval Research, ONR – 313, Arlington, VA	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT

Inverse Synthetic Aperture Radar (ISAR) provides full range detection and classification of sea and air based targets through two-dimensional range-Doppler imaging. The Naval Postgraduate School has developed a custom integrated circuit that can simulate false ISAR images in order to fool enemy ISAR platforms. To validate specific hardware choices within this design, this thesis explores the effect on image quality of an overflow occurring within the final 16-bit summation adder of this circuit. Three solutions to the problem of overflows are presented and analyzed. The logical extension of ISAR development, that of three-dimensional target imaging, is next presented through the discussion of 3D monopulse radar, 3D interferometric ISAR, and a 3D, three receiver ISAR. The relative strengths of each approach are compared, along with both MATLAB and X3D software models created for one specific 3D ISAR implementation. Through the superposition of 2D ISAR images it is shown how 3D ISAR images may be created. Moreover, emphasis is placed on using this knowledge to both enhance current 2D ISAR techniques and to modify the false-target chip to handle 3D ISAR return signals. The thesis concludes with a study of Non-Uniform Rational B-Splines, through which the X3D software model was created.

14. SUBJECT TERMS Digital Image Synthesizer, DIS, ISAR, 3D ISAR, Adder Overflow Analysis, NURBS, X3D	**15. NUMBER OF PAGES** 150
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

THREE DIMENSIONAL IMAGE SYNTHESIS: THEORY AND APPLICATION

Charles N. Adams IV
Ensign, United States Naval Reserve
B.S.E., University of Pennsylvania, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2003

Author: Charles N. Adams IV

Approved by:

Phillip E. Pace
Thesis Advisor

Don Brutzman
Co-Advisor

John Powers
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Inverse Synthetic Aperture Radar (ISAR) provides full-range detection and classification of sea-based and air-based targets through two-dimensional range-Doppler imaging. The Naval Postgraduate School (NPS) has developed a custom integrated circuit that can simulate false ISAR images in order to fool enemy ISAR platforms. To validate specific hardware choices within this design, this thesis explores the effect on image quality of an overflow occurring within the final 16-bit summation adder of this circuit. Three solutions to the problem of overflows are presented and analyzed.

The logical extension of ISAR development, that of three-dimensional target imaging, is next presented through the discussion of 3D monopulse radar, 3D interferometric ISAR, and a 3D, three-receiver ISAR. The relative strengths of each approach are compared, along with both MATLAB and Extensible 3D (X3D) Graphics software models created for one specific 3D ISAR implementation. Through the superposition of 2D ISAR images it is shown how 3D ISAR images may be created. Moreover, emphasis is placed on using this knowledge to both enhance current 2D ISAR techniques and to modify the false-target chip design to handle 3D ISAR return signals. The thesis concludes with a study of Non-Uniform Rational B-Splines, through which the X3D software model was created.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my advisors, Phillip Pace and Don Brutzman, for their guidance throughout the course of this thesis. Moreover, thank you to all those who have supported me throughout my time at NPS. In particular, thank you to my family for their thoughts and prayers and to God for His strength and wisdom.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis provides both model validation for the current Digital Image Synthesizer (DIS) design and research into the possibility of creating three-dimensional Inverse Synthetic Aperture Radar (ISAR) images from current two-dimensional (2D) images. The DIS is a hardware-implemented method of generating false ISAR return signals based upon realistic ship-type coefficients. The goal of this system is to fool traditional ISAR detection platforms such as the AN/APS-137 radar aboard surveillance aircraft.

Model validation is accomplished through the systematic analysis of hardware design choices leading up to the subsequent overflow of the 16-bit final summation adder within the DIS. This final-summation adder sums the modulated phase samples from up to 512 range bins within the DIS in order to create the desired return signal. Unfortunately, adder overflow degrades overall image quality by inadvertently toggling the sign bit of the adder. The most cost-effective solution to this problem is to modulate the overall gain of the signal by reducing the bit-level precision of the binary gain shifter. Although some dynamic resolution is lost, image correlation to within 0.9898 of the ideal image is achieved. Further model validation is achieved through the creation of software within MATLAB to both generate 2D range-Doppler images and extract coefficients from realistic target data. These coefficients can then be used to run the DIS.

A foundation for three-dimensional ISAR is laid through the discussion of previous work in the field including 3D monopulse radar, 3D interferometric ISAR, and a three-receiver approach to 3D ISAR. A comparison between the methods shows a common thread lies within the back scattering effects contained within the phase of coherent range bins for multiple 2D ISAR images. With this information, the third dimension of elevation can be constructed. Tools for visually analyzing 3D ISAR through the overlay of 2D images on a 3D geometric model within the graphical environment of X3D are presented. By examining feature correlation within the simulation, insight into both future 3D ISAR techniques and current 2D ISAR techniques is gained.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THREE DIMENSIONAL (3D) IMAGE SYNTHESIS OVERVIEW

The United States Military has remained a leading world power over the past fifty years. During this time, the fast-paced advance of technology has forced the military to be on the leading edge of that technology in order to remain one step ahead of potential enemies. Being on the technological forefront is especially important on the battlefield where soldiers' lives are at stake. The ability to both quickly and accurately detect and classify enemy targets provides the extra time needed to properly coordinate action against threats waiting just beyond the horizon. Synthetic Aperture Radar (SAR) has the capability to detect, image, and classify land-based targets. A modified approach to SAR, known as Inverse Synthetic Aperture Radar (ISAR) allows the classification of both sea-based targets and enemy aircraft through the use of two-dimensional range-Doppler imaging. While these current radar systems provide an advantage, their ever-increasing distribution poses a threat to future operations conducted by the U. S. and its allies.

Research is currently underway at the Naval Postgraduate School (NPS) to develop a custom integrated circuit (IC) that can accurately create the necessary ISAR return signature for a sea-based target in the attempt to fool enemy radar. A software simulation of the IC's hardware design was developed through the thesis research of Fernando Le Dantec [1] at the Naval Postgraduate School. This simulation models the entire false-target process from incoming radar chirp pulse to outgoing radar return signal. It serves as a foundation for the continued investigation of this thesis into IC design and concept validation.

Although the current software simulation has been shown to correctly model chip behavior, the integrity of the overall chip design must still be scrutinized in light of specific hardware implementation choices. One of these choices is the decision to use a binary power-of-two gain shifter to implement signal amplitude modulation by stored false target coefficients. This approach efficiently quantizes the gain into discrete levels. The disadvantage is that when the final modulated pulses are summed to create the

combined return signal, an overflow may occur downstream in the chip. The first part of this thesis involves an analysis of this overflow, providing detail on the source of the overflow, its effect on image quality, and two novel ways to deal with the problem. It is shown that acceptable image quality may be retained without significant loss in dynamic resolution by simply lowering the bit-precision and, thus, the signal gain of the power-of-two gain shifter.

The second part of this thesis deals with the output of the Digital Image Synthesizer (DIS) integrated circuit. Up until this point, the only verification of correct operation has been through the use of sample coefficients for one particular ship profile. However, new data is being acquired from the Naval Research Laboratory detailing an entire scenario in which a sea-based target approaches, completes a 180-degree turn and steams off relative to a fixed ISAR platform. The acquisition of this data allows for both the creation of 2D range-Doppler images based on this received data and for the extraction of gain coefficients from the data by which the DIS model may be run. The output of the two models can then be compared to verify the accuracy of the DIS.

The final part of the thesis deals with the exploration of the logical extension of ISAR systems. While competing countries develop their own ISAR and counter-ISAR technologies, the foundation will already be laid for the United States to take the next step into 3D ISAR imaging. The premise for current research in 3D ISAR is laid, alongside a discussion of both the implementation of a 3D ISAR and the benefits of doing so. An X3D model is presented in order to visually gain a concept of the advantages gained from simply overlaying multiple 2D images on a 3D ship model. This approach allows for insight into new features for and problems with current 2D ISAR implementation. The section concludes with an emphasis on how the current 2D DIS may be converted to a 3D DIS for improved counter ISAR capability.

Through the creation of the tools necessary to conduct research for this thesis, the author has provided significant advances in terms of providing Non-Uniform Rational B-Spline (NURBS) support to the X3D simulation platform used in developing the 3D ISAR model. This research is included in Chapter 6 as part of the three-dimensional visualization tools discussion.

B. PRINCIPAL CONTRIBUTIONS

Through the analysis of possible hardware overflows within the final 16-bit summation adder of the Digital Image Synthesizer (DIS) chip, certain design problems are uncovered that significantly degrade the overall quality of the resulting 2D range-Doppler images. These problems may be exploited by enemy forces to differentiate between real and false return signals, thus invalidating the value of the DIS. However, by lowering the overall gain of the return signal through lowering the bit-quantization of the power-of-two gain shifter in the DIS, the problem of overflows in the final summation adder is resolved without significant loss in dynamic signal range.

Two-dimensional ISAR has the disadvantage that it cannot resolve scattering elements within the elevation dimension of a target. For this reason, it is difficult to produce a three-dimensional ISAR image by conventional techniques. Through exploring recent approaches to producing 3D ISAR, and subsequently developing a software-based model to superimpose 2D ISAR images on a 3D target, the foundation for future work in implementing advanced 3D ISAR algorithms is laid. Information is also presented on how the current DIS model must be modified to support the creation of return signals that include the extra back-scattering information needed to resolve the third dimension of elevation.

Finally, a significant contribution is made to the X3D graphics community through the implementation of Non-Uniform Rational B-Splines (NURBS). NURBS is a method of mathematically describing smooth curves and surfaces through a minimal set of control points. They have been a part of the X3D specification since 1997 but until now have not been uniformly implemented for general-purpose use within the X3D environment. This contribution includes a plug-in for the industry standard graphics modeling package, Maya, that allows for complex NURBS models to be exported to X3D format while maintaining their full resolution.

C. THESIS OUTLINE

Chapter II gives an introduction to high-resolution radar, including the general principles of radar aperture and range resolution. Important mathematical equations are developed to describe each aspect presented. Emphasis is placed on both Synthetic Aperture Radar (SAR) and Inverse Synthetic Aperture Radar (ISAR). The unclassified specifications for the AN/APS-137 and AN/APS-147 radar platforms are presented as well.

Chapter III describes the current work that has been done on the Digital Image Synthesizer chip to create false ISAR images. A development of important equations is given for the complete DIS process, including the intercepted chirp pulse signal, range and azimuth compression, gain and phase modulation, and finally signal summation and output. The corresponding MATLAB model of the DIS, based on the research of Fernando Le Dantec [1], is briefly presented.

Chapter IV follows the development given in Chapter II by examining the specific hardware choices pertaining to the possible overflow within the final 16-bit summation adder. Reasons for the overflow are discussed along with three approaches to the issue and an analysis of each approach. Simulated ISAR images are presented to support the analysis.

Chapter V describes in more detail the creation of a 2D range-Doppler image and the coefficient extraction necessary to run the DIS from real ISAR data. The extension to the original software model of the DIS that allows for external data to be processed is presented.

Chapter VI examines three approaches to 3D ISAR including 3D monopulse radar, interferometric ISAR, and a three-receiver approach to 3D ISAR. Included are possible modifications that may be made to the current DIS to support one of these 3D ISAR methods. This discussion is followed by an overview of the X3D virtual environment software and the X3D model created to superimpose multiple 2D ISAR images on a 3D target. The chapter ends with a detailed discussion of the NURBS component of X3D as implemented for the 3D ISAR model.

Chapter VII gives a final summary and conclusion, connecting the principle contributions of each section and providing an overview of future work that may be done in the realm of 3D ISAR.

THIS PAGE INTENTIONALLY LEFT BLANK

II. SYNTHETIC APERTURE RADAR (SAR)

Current high-resolution radar has the capability to properly detect and process radar return signals from individual scattering elements in more than one dimension. Much more than simple range and bearing, this capability allows high-resolution images to be created of the target. Images such as the one shown in Figure 2.1 can be used to provide precise target identification at full range, thus posing a hazard to any naval operation that wishes to remain anonymous. The basic technology that allows for this kind of high-resolution imaging is Synthetic Aperture Radar (SAR). This chapter will present the fundamentals behind high-resolution radar including SAR, with particular emphasis on the operating mode known as Inverse Synthetic Aperture Radar (ISAR). ISAR will be shown to expand the functionality of SAR by imaging moving targets such as planes and ships in contrast to the land-based targets of SAR.

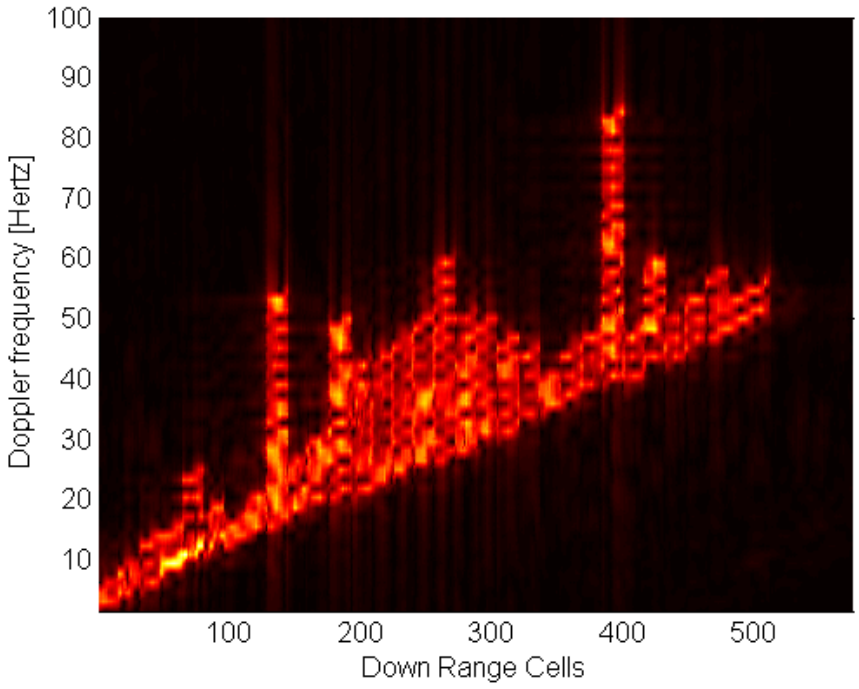


Figure 2.1 2D Range-Doppler Image for Infinite (Ideal) Resolution.

A. GENERAL RADAR PRINCIPLES

The direction directly along the line of sight of a radar beam is known as the slant range. Following the development in [2], the fundamental equation for resolving scattering elements in the slant-range for high-resolution radar is

$$\Delta r_s \approx \frac{c}{2\beta} \tag{2.1}$$

where c is the speed of light and β is the bandwidth of the radar. Specifically, high-resolution radar refers to systems transmitting signals within the wideband range. When a radar signal is transmitted, the received radar echo-power under free space propagation conditions for a target uniformly illuminated by an antenna beam is represented as

$$s_1 = \frac{P_t G_t}{4\pi R^2} \tag{2.2}$$

where

P_t is the RF power in watts from the transmitter,

G_t is the transmitter antenna gain, and

R is the target range in meters.

The effective aperture of an antenna is defined as the ratio of received power to the power density of the incident wave. A general equation relating the gain of an antenna to its effective aperture is given by

$$G = \frac{4\pi}{\lambda^2} A \tag{2.3}$$

where λ is the wavelength of the transmitted signal and A is the effective aperture of the antenna. Notice how a larger aperture produces a larger gain. For a single antenna used to transmit and receive pulses, the gain of the transmitter equals the gain of the receiver ($G_t G_r = G^2$) and Equation 2.3 becomes

$$S = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{2.4}$$

where σ is the radar cross-section of the target.

In the presence of external noise and signal losses, the ability of a radar system to detect the return pulse is defined as its sensitivity, modeled by

$$S_r = kT_s \beta_n F_n (S/N)_{in} \tag{2.5}$$

where

k is Boltzmann's constant,

T_s is the receiving noise temperature,

β_n is the receiving noise bandwidth,

$F_n = \frac{(S/N)_{in}}{(S/N)_{out}}$, is the ratio of input signal-to-noise to output signal-to-noise.

Combining Equations 2.4 and 2.5 yields the exact form for representing the range of a single antenna in terms of signal to noise ratio and in the presence of system losses

$$R = \left[\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 kT_s \beta_n F_n (S/N)_{in}} \right]^{1/4}. \tag{2.6}$$

These equations will be the basis for developing the formulas to model high resolution SAR and ISAR.

B. (SAR) BACKGROUND

The term cross-range refers to the distance perpendicular to the line of sight, or slant-range, of the radar. Cross-range is also referred to as azimuth resolution. A real aperture radar relies upon the physical dimensions of the antenna to dictate the maximum cross-range resolution of the radar beam. This cross-range resolution is approximately equal to the 3-dB beamwidth of the antenna times the range [3], or

$$\theta_{3dB}[\text{radians}] \cong \frac{\lambda}{L} \tag{2.7}$$

where L is the length of the antenna. Thus to obtain high resolution, either a very short wavelength, or a very long antenna is needed. Due to atmospheric limitations, the smallest achievable wavelength for long-range mapping is around 3 cm. The idea of using a very long antenna is simply not feasible on most surveillance aircraft. For example, it would take a 16-foot antenna operating in the X Band region to track small craft at distances of 10 nautical miles [3].

In response to this need to obtain higher resolution without increasing the antenna length or decreasing the wavelength, the Synthetic Aperture Radar concept was developed. The general theory behind SAR is to use a small, side-looking antenna capable of transmitting a narrow pulse whose beamwidth is approximately half that of a traditional aperture. The antenna is mounted on a forward moving aircraft such that the integration of coherent pulses creates a cross-range dimension that would correspond to what a real, and much larger, aperture would produce. A finite length of time is therefore required to synthesize the return pulses and create the desired target image.

The three most common operating modes that characterize SAR are stripmap, scan, and spotlight. Figure 2.2 shows each of these modes.

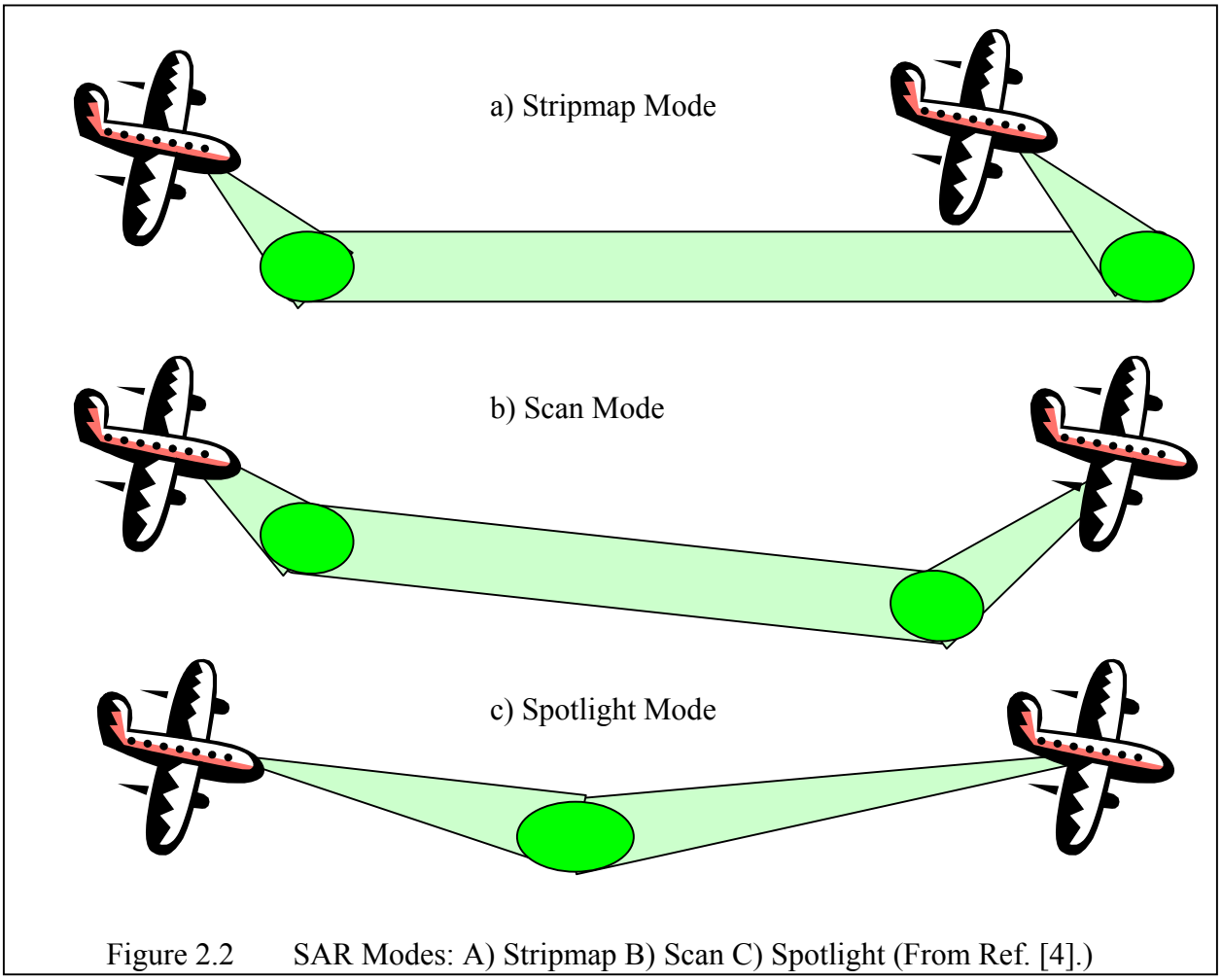


Figure 2.2 SAR Modes: A) Stripmap B) Scan C) Spotlight (From Ref. [4].)

1. SAR Modes

In stripmap mode, Figure 2.2a, a fixed side-looking antenna is attached to the aircraft. This antenna has a real, physical aperture. The radar beam uniformly illuminates a strip of ground as the aircraft traverses the target. The velocity of the aircraft combined with the pulse repetition time determine the integration time. In this configuration, points closer to the radar require a shorter integration length than points farther away from the radar. As a result, the maximum resolution in cross-range for strip-map mode is in effect one-half of what a real aperture's cross range would be. Moreover, since the target dwell time, or time the target is within the radar beam, is relatively short, image resolution typically suffers. [4]

Scan mode is represented in Figure 2.2b. Returned pulses are integrated while the target is scanned by the radar beam in cross-range. Here, the resolution is limited by the rate at which the antenna scans the target in addition to the physical beamwidth of the pulse. By scanning across a target while the aircraft moves past it, the synthetic antenna beam is effectively sharpened over traditional stripmap mode.

The third operating mode, spotlight SAR, is illustrated in Figure 2.2c. Here, the antenna beam stays focused over the target while the aircraft passes by at a specific azimuth angle. The cross-range resolution of spotlight mode is limited by the target dwell time and radar beamwidth. This mode provides the sharpest resolution as the radar can take advantage of a fairly long target dwell time. Typically, scan-mode will be used until a target is spotted at which point spotlight mode is switched on. Spotlight mode will be the assumed mode for the remainder of this thesis as the target will already have been found. The cross-range resolution for spotlight SAR is given as [2]

$$\Delta r_c \approx \frac{1}{2}\frac{\lambda}{\psi} \tag{2.8}$$

where ψ is the integration angle.

Current SAR and ISAR platforms also use what is known as focused radar beams. The difference between a focused and unfocused beam is that the focused beam takes into account the quadratic phase error produced by the motion of the radar past its target. The movement of the aircraft platform away from the target is also generally taken into account when integrating return pulses. By focusing the radar beam, the effective cross-range resolution of the radar is increased. According to Wehner [2], the half-power resolution for unfocused SAR in strip-map mode is

$$\Delta r_c \approx 0.5\sqrt{R\lambda} \tag{2.9}$$

where the resolution degrades as the square of both range and wavelength. In contrast, the half-power resolution of focused SAR is

$$\Delta r_c \approx \frac{L}{2}. \tag{2.10}$$

By focusing the aperture, the resolution becomes independent of the range to the target.

C. INVERSE SYNTHETIC APERTURE RADAR (ISAR)

While SAR uses the motion of the radar platform to coherently integrate return energy from a generally land-based target, ISAR uses the motion of the target relative to an effectively fixed radar platform to integrate the return pulses and generate an image. This type of imaging works well for both ships and aircraft whose motion is typically somewhat constant, or repetitive. Referring back to the spotlight mode of SAR, the cross-range resolution is dependant on the width of the radar beam in which the target remains. ISAR is similar to spotlight mode where the target remains within the radar beam. However, it is not the width of the beam that matters, but the motion of the target and its dwell time within that beam.

To get a better understanding, consider an aircraft operating in spotlight SAR mode traveling along a circular path around a target at an integration angle ψ as in Figure 2.3a. Now fix the radar and move the target as in Figure 2.3b. The greater the motion of the target, the better the resultant image will be.

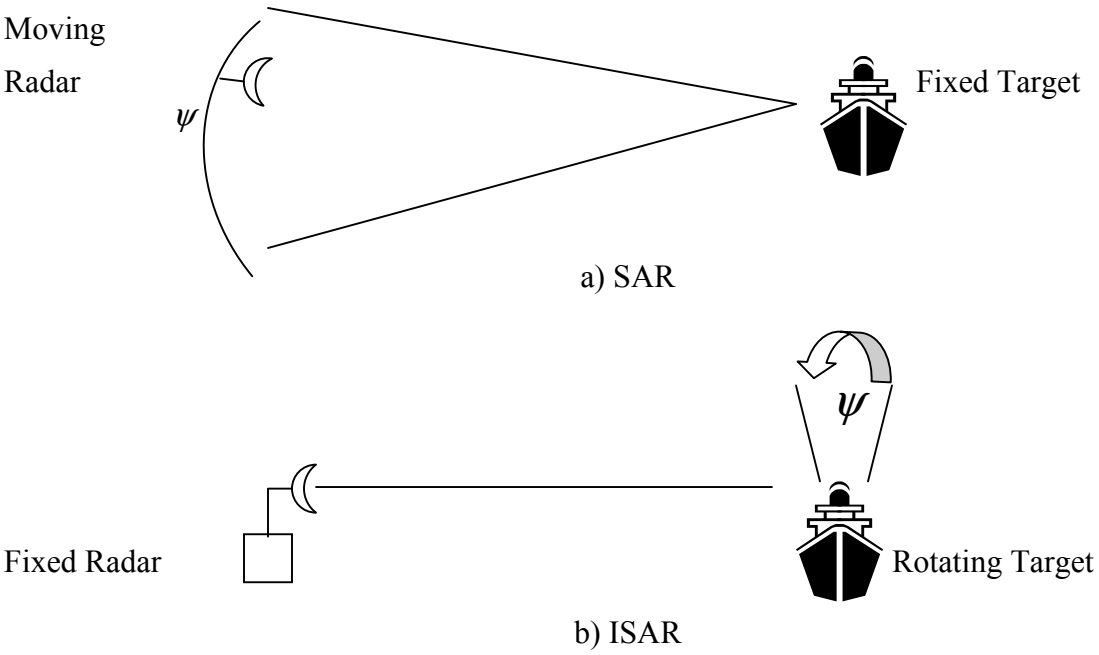


Figure 2.3 ISAR Radar Example (From Ref. [2].)

Similar to SAR, ISAR can use either a focused or unfocused beam as discussed previously. The optimum unfocused integration angle according to Wehner [2] is

$$\psi_{opt} = 1.2\sqrt{\frac{\lambda}{R}}. \tag{2.11}$$

The corresponding cross-range resolution is

$$\Delta r_c \approx 0.5\sqrt{R\lambda}. \tag{2.12}$$

For ISAR mode, focusing is accomplished by subtracting the two-way phase advance of subsequent scatters. In general, the motion of the target is not known ahead of time, and thus focusing ISAR requires special processing during imaging to detect and correct for target rotational motion. A focused ISAR has a half-power resolution of

$$\Delta r_c \mid_{3dB} = 0.44\frac{\lambda}{\psi} \tag{2.13}$$

and a Rayleigh resolution of

$$\Delta r_c = 0.5\frac{\lambda}{\psi}. \tag{2.14}$$

1. Chirp-Pulse ISAR Imaging

ISAR imaging is perhaps best understood as a series of range-Doppler profiles of the target being tracked. The range dimension is simply along the line of sight from the radar to the target. The Doppler dimension is based on integration of the return pulses from the rotation of the target. A combined return pulse may be broken down into a number of subsequent range cells, each containing a cross-range Doppler profile of the target at that range. Consider an integration time T during which the target rotates through an angle ψ. Assume there are n samples processed in both phase and quadrature-phase for each of N range profiles received during this time T. For each range cell, the Doppler magnitude is proportional to the reflectivity of the scatterer. This reflectivity is a product of both range and cross-range with the cross-range represented by an appropriate scaling factor based on the target's angular motion. According to Wehner

[2], the instantaneous Doppler-frequency shift of a single scatterer at a cross-range distance r_c and instantaneous velocity ω toward the radar is

$$f_D = \frac{2\omega r_c}{\lambda} \tag{2.15}$$

where ω is the instantaneous velocity and r_c is the cross-range distance.

Two scatterers in the same range cell, but different cross-range locations separated by a distance of δr_c will have a separation of frequencies equal to

$$\delta f_D = \frac{2}{\lambda}\omega\delta r_c \tag{2.16}$$

which can be rearranged as

$$\delta r_c = \frac{\lambda}{2\omega}\ \delta f_D\,. \tag{2.17}$$

If an ISAR has a Doppler-frequency resolution of Δf_D, then the cross-range resolution is given by

$$\Delta r_c = \frac{\lambda}{2\omega}\Delta f_D \tag{2.18}$$

where Δf_D is related to the coherent integration time by $\Delta f_D = 1/T$. Therefore, the cross-range resolution can be reduced to

$$\Delta r_c = \frac{1}{2}\frac{\lambda}{\psi}\,. \tag{2.19}$$

In order to resolve the target in slant-range, or along the line of sight of the radar, wideband chirp-pulse waveforms are typically used. The Rayleigh resolution for the chirp-pulse is

$$\Delta r_s = \frac{c}{2\Delta} \tag{2.20}$$

where Δ is the pulse bandwidth. The pulse falls within a slant-range window represented by

$$w_s = \eta_s \frac{c}{2\Delta} \tag{2.21}$$

where η_s is the number of complex samples delayed by Δt. The corresponding phase I and quadrature phase Q sampling rate must be at least D complex samples per second to meet Nyquist requirements for complete image formation. The slant-range integration length for a pulse duration T_1, which does not affect the sampling window, is $cT_1/2$, or the same as for SAR using chirp-pulse waveforms. The pulse repetition frequency needed for resolving reflections at a cross-range window of w_c is

$$\frac{1}{PRI} \geq \frac{2\omega w_c}{\lambda} \tag{2.22}$$

where *PRI* is the Pulse Repetition Interval and w_c is the cross-range window. This leads to the next expression which gives the minimum number of range-profiles needed for a given cross-range window,

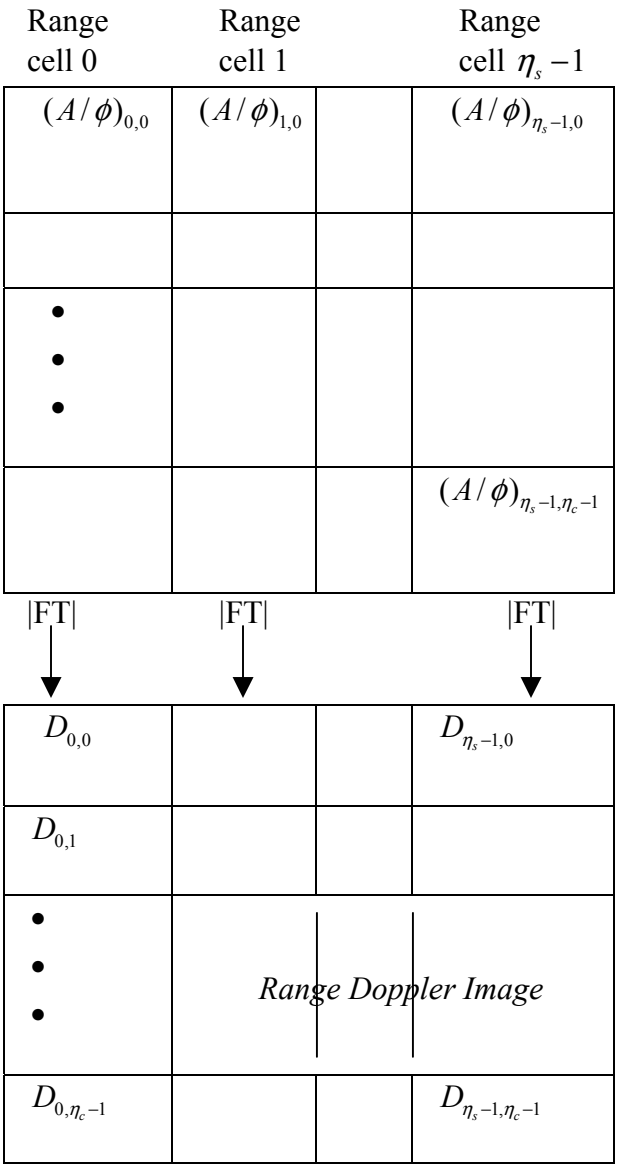
$$N \geq \frac{2\omega w_c PRI}{\lambda}. \tag{2.23}$$

The cross-range window itself can be represented with one profile for each of N chirp pulses as

$$w_c = N \frac{\lambda}{2\psi}. \tag{2.24}$$

To form a two-dimensional range-Doppler image of the target, the received pulses are typically down-converted to baseband, passed through a pulse-compression filter, and sampled in I and Q. The result is a time history matrix of the target's range-profile, separated into range bins as in Figure 2.4. The image can be produced by performing a Fourier Transform on the resulting time-history matrix by column. [2] The image is simply the magnitude of this matrix. An example of a two dimensional range-Doppler

image is shown in Figure 2.1. It is important to note that the range elements are not true representations of distance, but equally spaced samples in the slant-range dimension.



Symbol	Definition
$(A/\phi)_{l,k}$	Amplitude and phase in *l*-th range cell of the synthetic range profile from the *k*-th chirp pulse
$D_{l,j}$	Magnitude in *l*-th range cell and *j*-th Doppler cell of range-Doppler image
T	Image Frame Time $(T = NPRI)$

Figure 2.4 Image Processing of Chirp-Pulse Data (From Ref. [2].)

2. AN/APS-137

The AN/APS-137 represents the U.S. Navy's premiere maritime surveillance radar system capable of operating in both the SAR and ISAR mode at full range. It is an upgraded version of the AN/APS-116 radar, the main difference being the addition of ISAR capability. This radar is typically installed on P3 fixed-wing aircraft for anti-surface warfare purposes. The AN/APS-137 provides long-range ship classification and identification friend-or-foe capability through a parabolic roll and pitch stabilized antenna. At its core is a Receiver/Exciter/Synchronizer Processor (RESP) that can operate in both strip-map and spotlight modes. Among its more important tactical uses are periscope detection, surface surveillance and navigation. The AN/APS-137 has been actively deployed since 1998 to both U.S. and allied forces. Unclassified specifications for the AN/APS-137 are shown in Table 2.1 and Table 2.2 below. The contractor for the radar is Raytheon Company in McKinney, Texas. [5]

Specification	**SAR**	**ISAR**
Frequency	9.3-10.1 GHz	9.5-10 GHz
Average Power	350 W	230-500 W
Pulse-Width	12 μs	10 μs
Pulse-Width Compression	variable	6 or 13 ns
PRF	resolution dependant	500, 1000 Hz
Resolution	3, 9, 30 m	0.9, 1.8 m
Azimuth Coverage	360 degrees	
Receiver Sensitivity	92 dBm	
Receiver Noise	3.5 dB	

Table 2.1 AN/APS-137 Specifications(From Ref. [5].)

Target Radar Cross-sections(m^2)	Platform Altitude(m)	Range(km)
1	305	45
5	305	72*
10	305	72*
200	1524	161*
1000	3962	213
10000	6401	330*
*radar horizon limited		

Table 2.2 AN/APS-137 Specifications (From Ref. [5].)

3. AN/APS-147

The AN/APS-147 is the U.S. military's next generation of maritime surveillance radar. Similar to the AN/APS-137, it provides target classification, identification friend-or-foe (IFF), SAR and ISAR modes. However, advanced processing capability allows the AN/APS-147 to operate at substantially lower power levels than the AN/APS-137. Lower peak power waveforms combined with frequency agility mean a significantly lower probability of intercept by hostile forces when scanning medium to long-range targets. The AN/APS-147 is currently being integrated and tested aboard SH-60 helicopters. The full range of radar modes include SAR/ISAR imaging, periscope detection, long-range surveillance, weather detection and avoidance, all-weather navigation, and short-range search and rescue. Its increased resolution capability combined with its decreased power consumption make it an ideal choice for deployment in littoral situations. Detailed unclassified specifications for the AN/APS-147 are not available at this time. The contractor for the radar is Telephonics Corporation in Farmingdale, New York. [5]

D. SUMMARY

The equations developed through this chapter provide a general overview of high-resolution radar principles including antenna aperture, both real and synthetic, slant-range and cross-range dimensions, and their relation to SAR and ISAR operating modes. The ability to create a 2D range-Doppler image from ISAR return pulses forms the basis for this thesis, and the following development in Chapter III, relating to the Digital Image Synthesizer (DIS) integrated circuit chip designed to create false 2D ISAR return signals.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DIGITAL IMAGE SYNTHESIZER (DIS)

This chapter presents an overview of the Digital Image Synthesizer (DIS) design and the principles behind it. Emphasis is placed on the hardware implementation of the design, and on the overall process from intercepted chirp-pulse signal to the creation of an output waveform. Associated with the DIS is a MATLAB model created as part of the NPS Thesis of Fernando Le Dantec [1]. This software model works to validate the theory behind the DIS, as well as to provide a foundation for current research into 3D ISAR.

A. DIS OVERVIEW

Current wideband radar systems such as the AN/APS137 provide ISAR imaging techniques capable of identifying and thus classifying specific target types. In order to deceive such a radar, a coherent sequence of reflections, with proper delay, phase, and amplitude must be emulated. A radar pulse reflects off of many surfaces of a ship and at many different ranges to the ISAR platform. Each of these reflections must be taken into account when the false return is generated. The DIS is capable of digitally generating the necessary pulse-to-pulse phase coherence for all necessary reflective returns for nearly any target ship, up to the size of an aircraft carrier. To understand how, consider Figure 3.1.

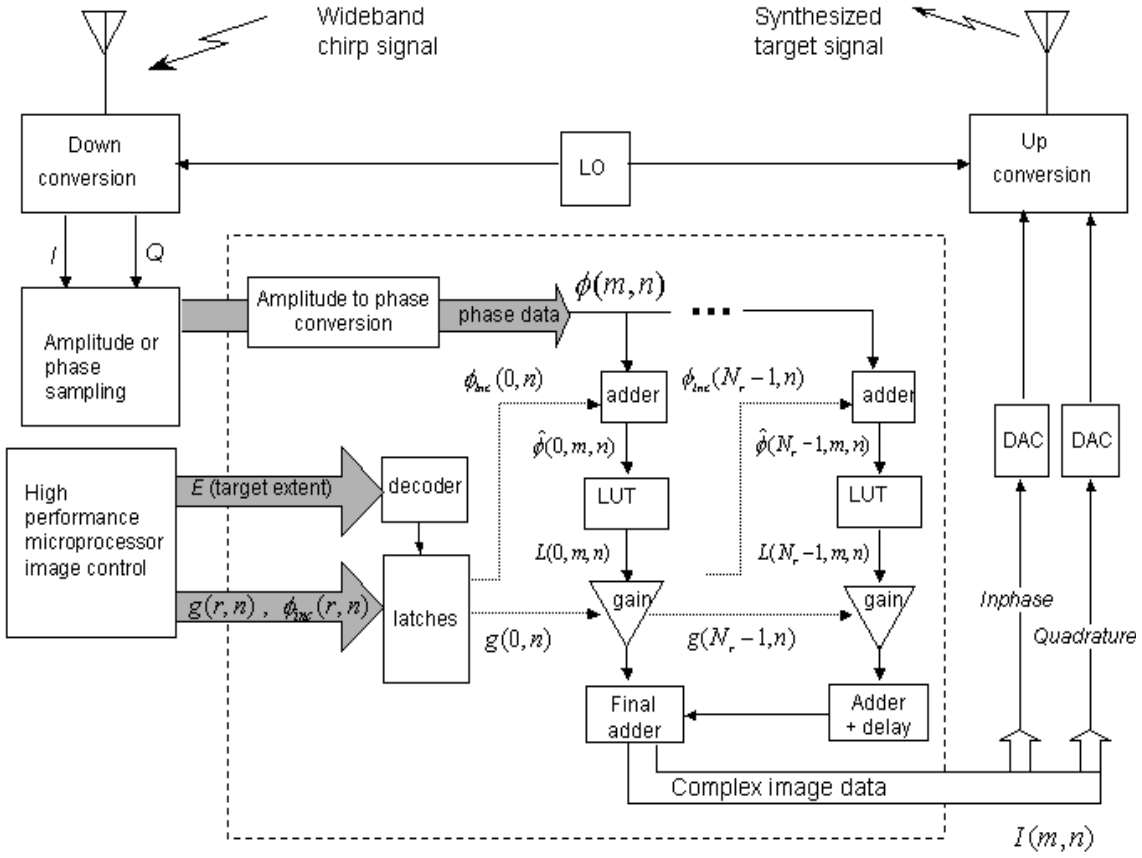


Figure 3.1 Digital Image Synthesizer Block Diagram (From Ref [6].)

Following the development in [6], an incoming wideband chirp-pulse radar signal is intercepted, down-converted to baseband frequency, and stored in Digital RF Memory (DRFM). The complex envelope of the intercepted chirp-pulse can be represented as

$$s(t) = \text{rect}\left(\frac{t}{\tau}\right) e^{j2\pi(f_d t + \Delta t^2/2\tau)} \tag{3.1}$$

$$\text{rect}\left(\frac{t}{\tau}\right) = \begin{cases} 1 & \left|\frac{t}{\tau}\right| < \frac{1}{2} \\ 0 & \left|\frac{t}{\tau}\right| > \frac{1}{2} \end{cases} \tag{3.2}$$

$$t = (m/f_s + nPRI) \tag{3.3}$$

where

τ is the pulse width in seconds,

f_d is the Doppler frequency between ISAR and receiving DRFM,

m is the sample number within chirp pulse of width τ,

f_s is the sampling frequency, must be greater than Δ,

n is the pulse number index from 0 to $N_p - 1$, and

N_p is the number of intercepted pulses stored in DRFM.

Equation 3.3 for t accounts for the N_p pulses stored in the DRFM. Combining Equations 3.1 and 3.3, and assuming that the PRI is constant for all N_p pulses, the phase samples from the incoming pulse can be represented by

$$\phi_o(m,n) = \left| 2\pi f_d (m/f_s + nPRI) + \frac{\pi\Delta}{\tau}(m/f_s + nPRI)^2 \right|_{2\pi} . \quad 3.5$$

These phase samples are further converted to integer values with k_p bit precision by

$$\phi(m,n) = \left\lfloor \phi_o(m,n) \Big/ \frac{2\pi}{2^{k_p}} \right\rfloor \quad 3.6$$

where $\lfloor x \rfloor$ is the largest integer value less than or equal to x. From this point, each pulse must be modulated in both amplitude and phase to match the desired false target, sent through a final summation adder, up-converted via a digital to analog converter and broadcast back to the victim radar. The strength of the false-return signal must be great enough to overcome the real-return signal but not so great as to overpower the radar system and act as a jamming device.

The DIS represents a custom integrated circuit chip that takes the down-converted samples, modulates them in amplitude and phase, and sums the digital signal to be up-converted. The remainder of this chapter focuses only on the DIS components and not upon the conversion steps before or after the signal is processed.

While the cross-range resolution of ISAR depends on the coherent return of subsequent pulses, the slant-range resolution depends on the number of range elements,

or bins, of which the return signal is composed. Similarly, the slant-range resolution of the DIS is also dependant on the number of range bins used to generate the false target. The current design includes 512 range bins on a single chip, each of which independently performs the modulation steps mentioned above on the incoming waveform. In order to emulate even larger targets, multiple DIS chips may be cascaded in parallel to achieve the necessary resolution.

As previously stated, the process in each range bin begins after sampling of the intercepted chirp-pulse is finished. Figure 3.2 describes the process by which the sampled pulse is modulated in both amplitude and phase, as well as the general layout of a single range-bin processor. The block diagram represents a four-stage pipelined process where the necessary gain and phase-modulation coefficients are obtained from pre-programmed off-chip memory. The extraction of these coefficients from real ISAR return signals for a given ship type is discussed in subsequent chapters.

Referring back to Figure 3.2, each range-bin modulator contains a phase adder, a look up table, a gain block, and a summation adder. Each of these components will now be discussed in further detail.

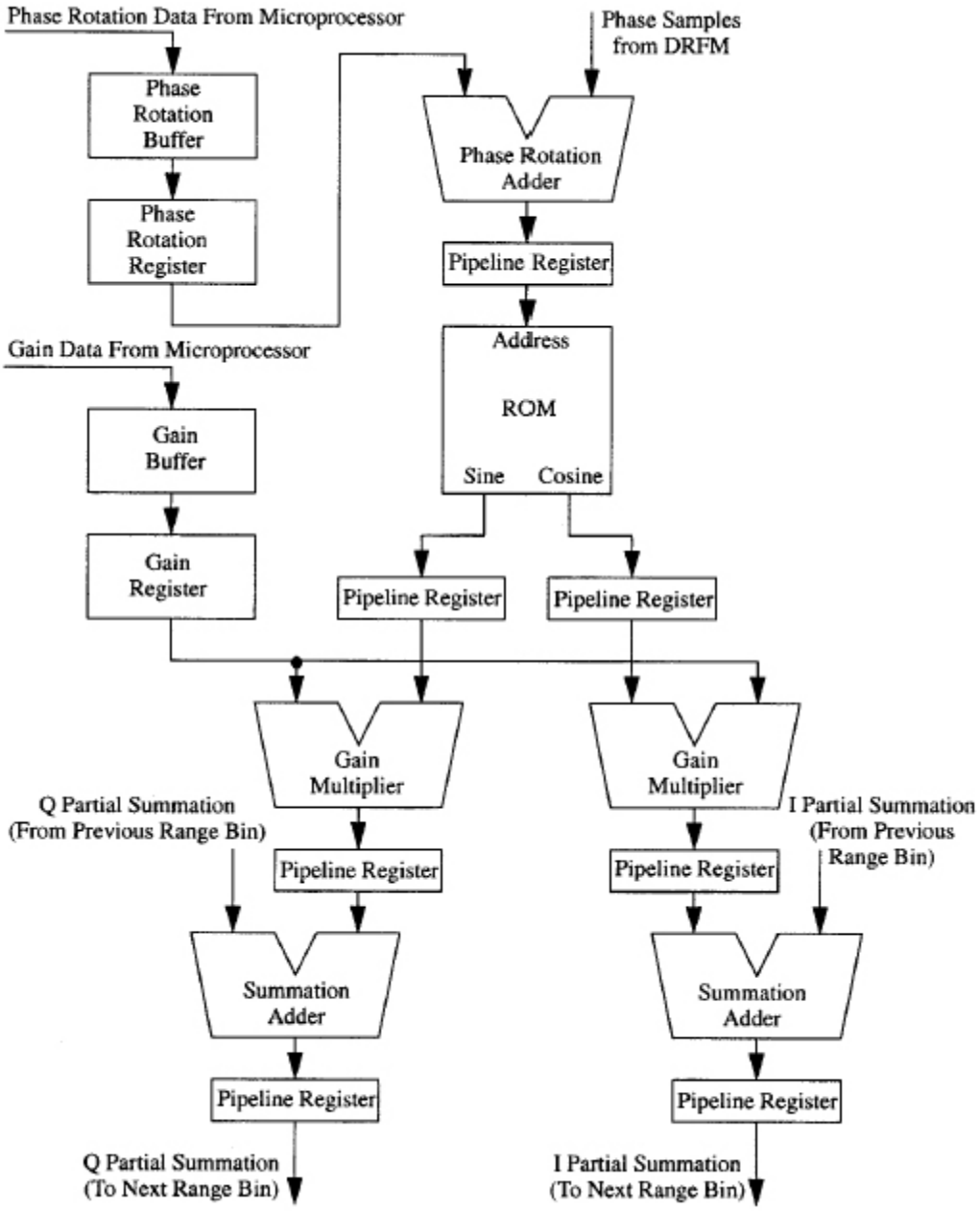


Figure 3.2 Range-Bin Architecture (From Ref. [7].)

The phase rotation is represented by a 4-bit unsigned number and as such is stored in 4-bit phase-rotation registers. By adding the phase coefficient to the current incoming pulse, the resulting phase is rotated by an amount equal to $\phi_{inc}(r,n)$ to provide a phase-adder output of [6]

$$\hat{\phi}(r,m,n) = \phi(m,n) + \phi_{inc}(r,n) \tag{3.7}$$

where r is the range-bin index. For each pulse-repetition interval (PRI), the phase can be rotated by a different coefficient. This allows for more realistic Doppler shifts within each range bin, as well as the ability to synthesize multiple targets at a time. Notice that, from Figure 3.2, both the phase and gain registers are double buffered to maintain high-speed throughput without switching delays.

At the output of the phase adder, the k_p-bit phase values are used by the lookup table to index a set of sine and cosine values in order to achieve the correct I and Q conversion for each pulse. The output of the lookup table is represented by

$$L(r,m,n) = e^{j(\phi(m,n)+\phi_{inc}(r,n))} \tag{3.8}$$

where $\phi(m,n)$ is the phase sample indexed by the m-th sample and n-th pulse and $\phi_{inc}(r,n)$ is the phase increment indexed by the r-th range bin and n-th pulse. From the lookup table, the new signal, modulated in phase, is modulated in amplitude by a power-of-two gain-shifter block, represented by

$$S(r,m,n) = 2^{g(r,n)} e^{j(\phi(m,n)+\phi_{inc}(r,n))} \tag{3.9}$$

where $g(r,n)$ is the number of bits by which to left shift I and Q.

A parallel array of multiplexers implements the gain shifter as shown in Figure 3.3. The gain shifter allows for modulation in both I and Q by up to 11 discrete power-of-two levels. In order to accurately recreate the necessary gain modulation, the coefficients must be normalized to fit within these discreet levels.

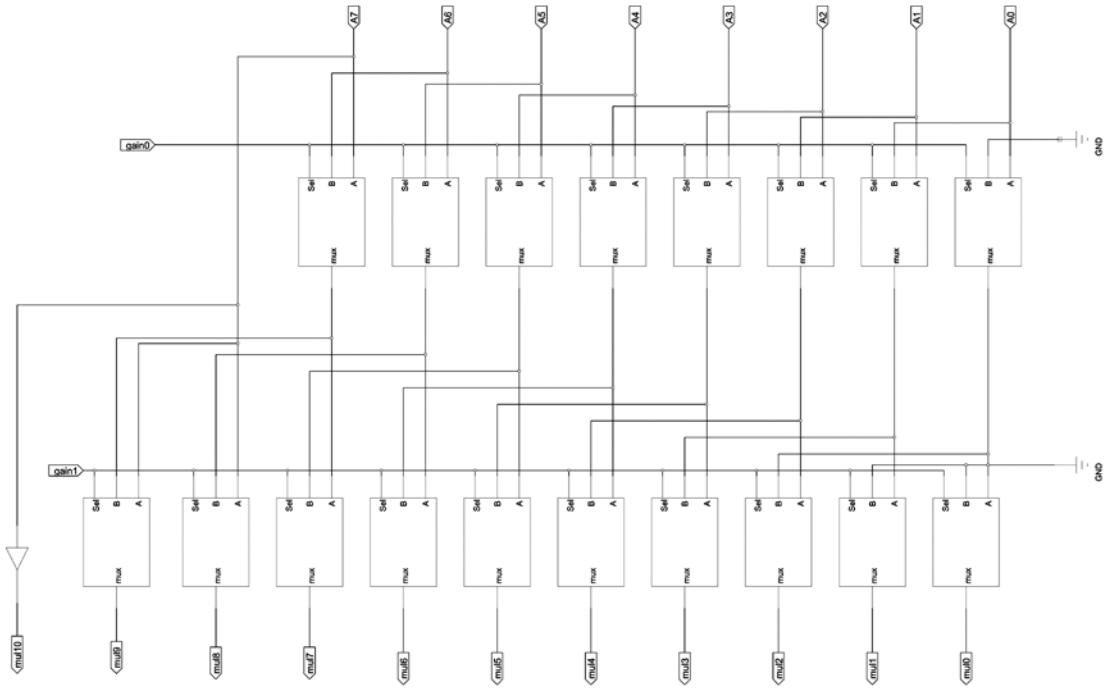


Figure 3.3 Gain Multiplier(After Ref. [7].)

In order to generate the output pulse, the modulated samples in each range bin must be summed with the previous range bin, as shown in Figure 3.4. The summation adder is implemented by a 16-bit two's-complement ripple-carry adder where the partial input of the current addition stage comes from the summation output of the previous range-bin stage. In general, up to 32 range bins may be summed without the possibility of an overflow due to the fact that both positive and negative values are being summed. However, cascading 512 or more range bins is needed for proper false target generation of large ships such as aircraft carriers. The next chapter deals with the underlying issues behind the possibility of having an adder overflow, the consequences of such an action happening, as well as methods to deal with this overflow.

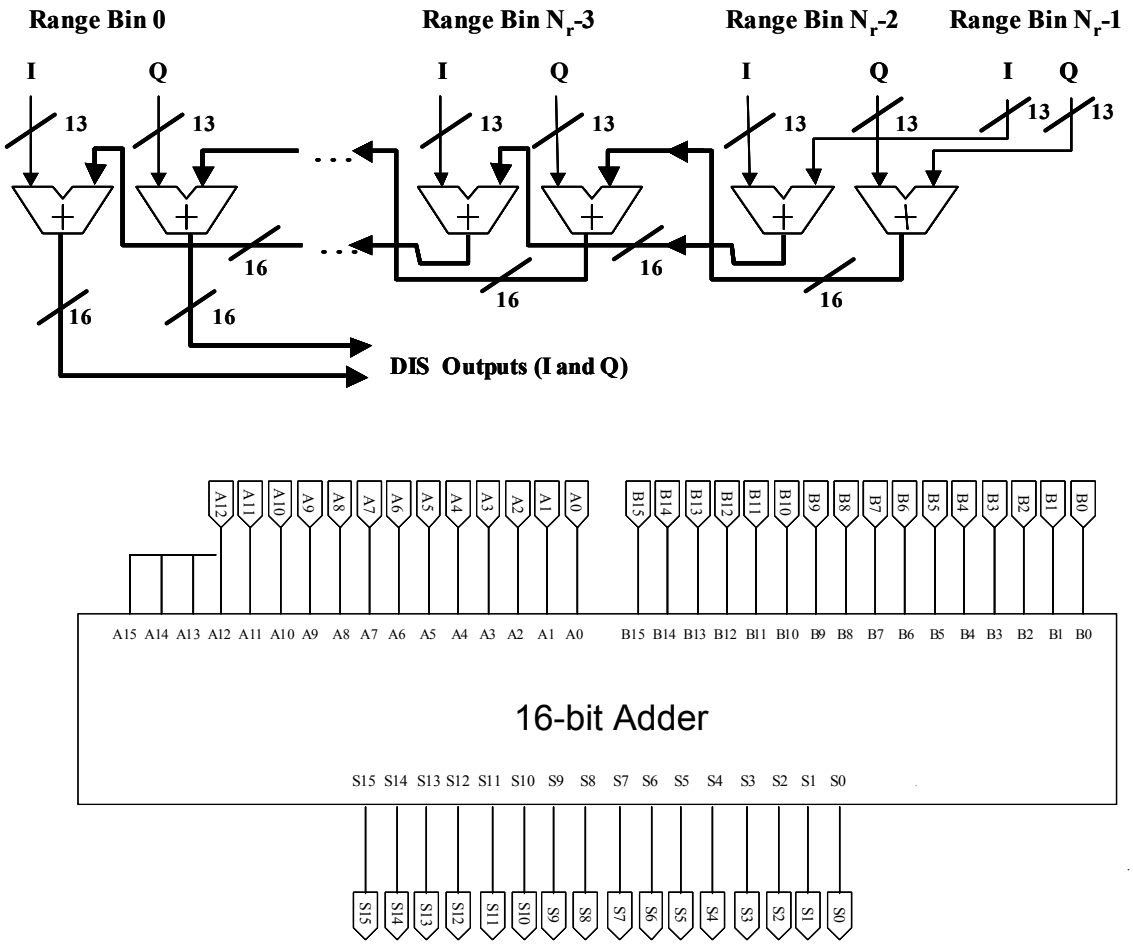


Figure 3.4 16 Bit Summation Adder (From Ref. [1].)

The output pulses from the DIS, created through the superposition of N_r copies of the pulse are

$$I(m,n) = \sum_{r=0}^{N_r-1} 2^{g(r,n)} e^{j(\phi(m-r,n)+\phi_{inc}(r,n))}. \qquad 3.10$$

Each of these copies is modulated in gain by $2^{g(r,n)}$ and in phase by $\phi_{inc}(r,n)$. Moreover, each pulse is appropriately delayed via the summation adder. These pulses are subsequently processed through a digital-to-analogue converter (DAC) and up-converted onto the carrier frequency for retransmission. The resultant signal is received by the initial ISAR platform, which processes it to form a range-Doppler image of the false

target. Convinced that the detected vessel is not a threat, the surveillance aircraft continues on.

B. MATLAB SIMULATION

Verification of the DIS and overall chip behavior is done through a mathematical model implemented in MATLAB by a previous NPS student, Fernando Le Dantec [1]. Within his thesis is a detailed explanation of the code behind the DIS as well as a copy of the entire code base. Since this thesis builds upon his research and code-base, a short summary of his model is presented here. For an exhaustive analysis, please see the reference to the original thesis.

The MATLAB simulation loads a sample target's Doppler template through execution of the file extract_v5.m. The template is as shown in Figure 3.5 and contains the main target scatterers arranged by range, r, and Doppler shift, $f(r,d)$ where d represents the cross-range location in terms of Doppler resolution.

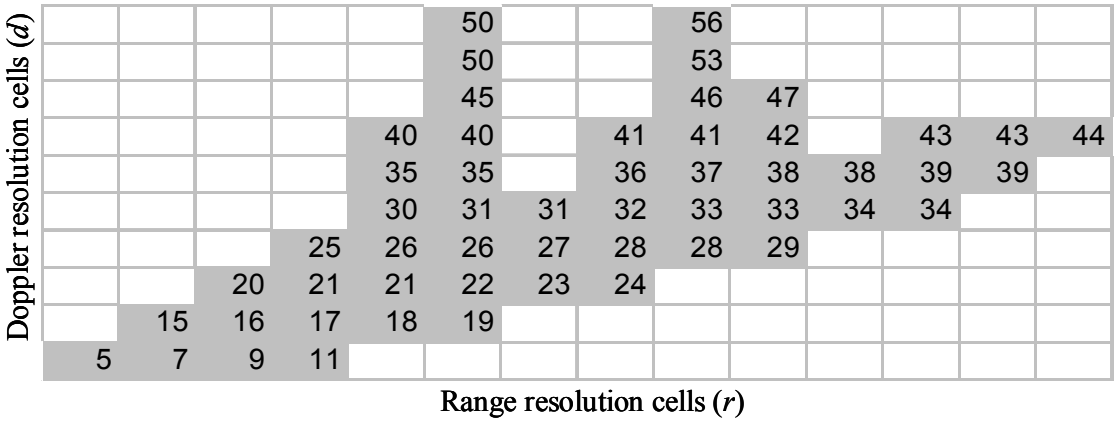


Figure 3.5 Range-Doppler Template (From Ref. [1].)

By loading different templates into the simulation, targets of sizes ranging from 2 to 512 range bins may be produced. Each version of the sample target is simply scaled from the previous version and represents the same perspective of the same ship. Chapter IV describes how the simulation can be used to compute false target images based upon the acquisition of realistic sample data. Regardless of the input data or coefficients, however, the process is the same.

Given a sample image, the simulator will first create a matrix of Doppler frequency values with a common magnitude. Their phase is determined based upon the Doppler shift contained within the frequency matrix. The general complex representation of the scattered return from the target is [6]

$$T(r,d,t) = A(r,d)e^{-j2\pi f(r,d)t} \qquad 3.11$$

where

d is the Doppler scatterer index,

$f(r,d)$ is the range-Doppler frequency configuration, and

$A(r,d)$ is the scattering amplitude.

A separate two-dimensional matrix, targetSum, [1] is created and represents a collection of range profiles for each of N_p pulses and is equivalent to the range profiles formed after the ISAR range compression output. From this matrix, the magnitudes of the range profiles are normalized into eleven levels corresponding to the binary gain shifter described above. These eleven levels are within the range of 2^0 to 2^{10}, each left shift representing multiplication by powers of two.

To extract the necessary phase coefficients, the simulation uses MATLAB's angle() function on the targetSum array and normalizes the result from 0 to 31.99. For each scatterer, a phase increment is calculated with respect to the scatterer of the previous Doppler location, starting at zero for the first location. The relative phase increments are stored in their own matrix as phase coefficients. [1]

Once the coefficients for magnitude and phase are obtained, the program simulates ISAR range compression using a cross-correlation algorithm and Fourier transforms. For each range bin, the sample chirp-pulse signal is modulated by the stored phase and magnitude coefficients, then given the proper delay and summed with the previous range bin. After range compression, azimuth compression is simulated to create a final matrix whose intensity values represent the target range-Doppler image as shown in Figure 2.1. The whole process is diagramed in Figure 3.6. Output images are

compared against the infinite resolution case in order to determine the correlation between real ISAR images and what is output by the DIS.

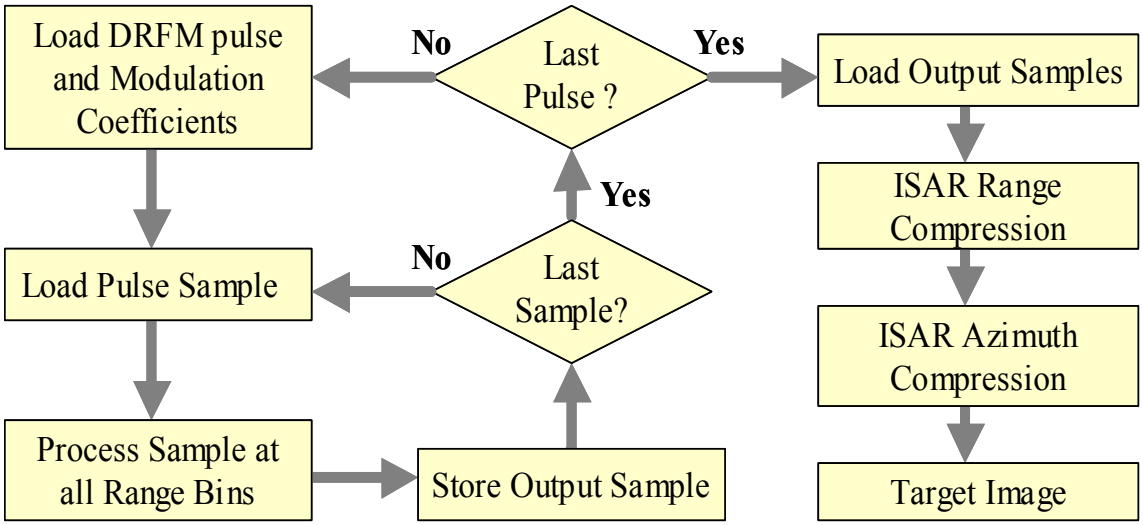


Figure 3.6 Flow Diagram of simhwchk_v5.m (From Ref. [1].)

D. SUMMARY

In this chapter, the mathematical basis underlying the DIS is developed along with a general introduction to its hardware implementation. Signal processing begins with the intercepted chirp-pulse, and continues through down-conversion, amplitude and phase modulation, coherent range-bin summation, and ultimately up-conversion. At the most basic level of understanding, the DIS recreates what should be a realistic copy of an ISAR return signature based on real-target coefficients. However, due to both the digital nature of the DIS as well as the necessary tradeoffs between hardware efficiency and signal accuracy, what is instead outputted is a high-level approximation of the desired return signature. Design choices that play a significant role in governing both efficiency and accuracy include implementation of the gain multiplier and final summation adder. Chapter IV explores the specific relationship between these hardware components and the resulting output image quality in light of realistic operating conditions.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ADDER OVERFLOW ANALYSIS

This chapter presents a detailed discussion of the hardware implemented gain shifter and final summation adder of the DIS. Due to implementation considerations within these blocks, the possibility exists, when simulating false images of large targets, for an overflow to occur in the final summation adder. This overflow is detrimental to the overall image quality and thus may become a major issue when validating the accuracy of the DIS chip. Moreover, the overflow may present characteristics of images that, when exploited, might be used to determine that the target is false, thus invalidating the value of the entire DIS system. Examples of the overflow are shown for different size images, culminating with a 512-range bin target. In response to this problem, three ways to deal with the overflow are presented along with the relative strengths and weaknesses of each design. The first method involves setting a flag when the overflow occurs but allowing the image to be degraded. The second method includes capping each range-bin summation at a maximum value instead of allowing the overflow to wrap around. The third method uses a system whereby the maximum allowed gain shift is reduced from 2^{10} down to 2^9 or even 2^8 to reduce the overall gain of the waveform. It will be shown that this last method provides the most effective, and least costly, method as far as system redesign is concerned.

A. POWER-OF-TWO GAIN SHIFTER

After the incoming chirp-pulse is down-converted to baseband frequency and sampled in phase I and quadrature phase Q, each component is rotated by phase coefficients and processed through a lookup table to be in the form [6]

$$L(r,m,n) = e^{j(\phi(m,n)+\phi_{inc}(r,n))}. \tag{4.1}$$

It is at this point that, in each range bin, the I and Q components all have a uniform, constant magnitude. In order to correlate their magnitude with that of the real return pulse of the false target, Equation 4.1 must be multiplied by the correct amplitude coefficient. There are many ways to implement the required gain multiplier. A true

arithmetic multiplier would require a significant number of transistors, incur detrimental delays, and occupy a large portion of real estate on the fabricated chip. With two gain multipliers per range bin, one for I and one for Q, and 512 range bins, it becomes impractical to use a full-scale array multiplier. Instead, the gain-modulation coefficients are normalized to eleven discrete levels representing gain multiplication of 2^0 to 2^{10}, or 1 to 1024 times the original magnitude. Using a gain shifter instead of a full arithmetic multiplier is not an issue since a dynamic range of approximately 60 dB is still realized using four bits to control the eleven power-of-two shifts. Moreover, depending on how the original magnitudes are normalized to these eleven levels, extra precision can be allotted to the areas of the signal that require it. One non-linear example of a gain-quantization scheme is shown in Table 4.1. [1] The currently implemented 11-bit gain shifter is shown in Figure 3.3.

Magnitude Ranges	**Number of Shifts *g(r,n)***	**Effective Gain $2^{g(r,n)}$**
0.8 to 1.0	10	1024
0.4 to 0.8	9	512
0.2 to 0.4	8	256
0.1 to 0.2	7	128
0.05 to 0.1	6	64
0.025 to 0.05	5	32
0.0125 to 0.025	4	16
0.0063 to 0.0125	3	8
0.0032 to 0.0063	2	4
0.0016 to 0.0032	1	2
0 to 0.0016	0	1

Table 4.1 Gain Quantization (From Ref. [1].)

Regardless of how the gain multiplier is implemented, the larger the allowed multiplication factor is, the larger the final summation is at the output of each range bin and thus the greater the chance for an overflow. Notice that modulating the phase of the false target signal does not present the possibility of an overflow since the addition of phase coefficients serves only to rotate the phase through a maximum angle of 360 degrees.

B. 16-BIT SUMMATION ADDER

The output of the gain shifter for each pulse in each range bin is a 13-bit two's complement number. One sign bit, ten integer bits, and two decimal bits form binary values ranging from –1024 to 1023.75. To create the final return signal, the scattering elements from each range bin must be summed. This is accomplished through a 16-bit two's complement ripple-carry adder. Each range bin has one adder for each of its component I and Q parts. Starting at the first range bin, one input to the adder comes from the sign-extended 13-bit result of the binary gain shifter. The other input comes from the 16-bit sum of the previous range bin. For the first range bin, the secondary input is zero. [7] With each subsequent clock pulse, the final summation proceeds to the next range bin where it is added to the output of that stage's gain shifter. By rippling through the range bins on a clock-by-clock basis, the proper delay is inserted for scattering elements at varying ranges. When the final summation is completed, the resultant signal is ready for the digital-to-analog converter (DAC) and transmission back to the victim ISAR.

In two's complement arithmetic involving k bits, the maximum signed range is from -2^{k-1} to $2^{k-1}-1$. For 1 sign bit, 13 integer bits, and 2 decimal bits, this range translates to –8192 to 8191.75. When an overflow occurs, the result is a spill-over into the sign bit of the two's complement number, thus changing a positive number to negative, or a negative number to positive. When the overflow occurs early in the ripple-carry adder, the error will propagate through every stage and every range bin where it is compounded by future additions. The next section shows the effects of this overflow on image quality.

C. OVERFLOW SIMULATION RESULTS

The MATLAB simulation of the DIS was run for identical sample target images, employing both 32 range bins and 512 range bins. A 512-range bin infinite-resolution test case was also run to produce what an ideal image looks like. Intermediate steps of 64, 128, and 256 range bins were not considered for three reasons. First, overflow problems do not present themselves unless significantly high resolutions are used. Second,

simulation times for computing individual stages on a software-implemented design in MATLAB are very time-consuming. For showing results from a situation that does not involve overflow, 32 range bins provides an acceptable, yet fast solution. Finally, the majority of false targets occurring in real-world tactical operations must be at least 512 range bins in size to provide realistic return data. Since each DIS chip has up to 512 range bins, if the problem can be solved within one chip, then cascading multiple DIS chips for even larger targets will not be an issue since each chip uses its own summation adders. Furthermore, the results from each chip can be externally summed and modulated if required before being up-converted.

1. 32 Range bins

The first set of figures represents simulation outputs at various points along the DIS process from intercepting the wideband chirp-pulse to displaying the 2D range-Doppler image of the signal outputted by the DIS. The intercepted ISAR chirp-pulse, as represented by Equation 3.1 is shown in Figure 4.1.

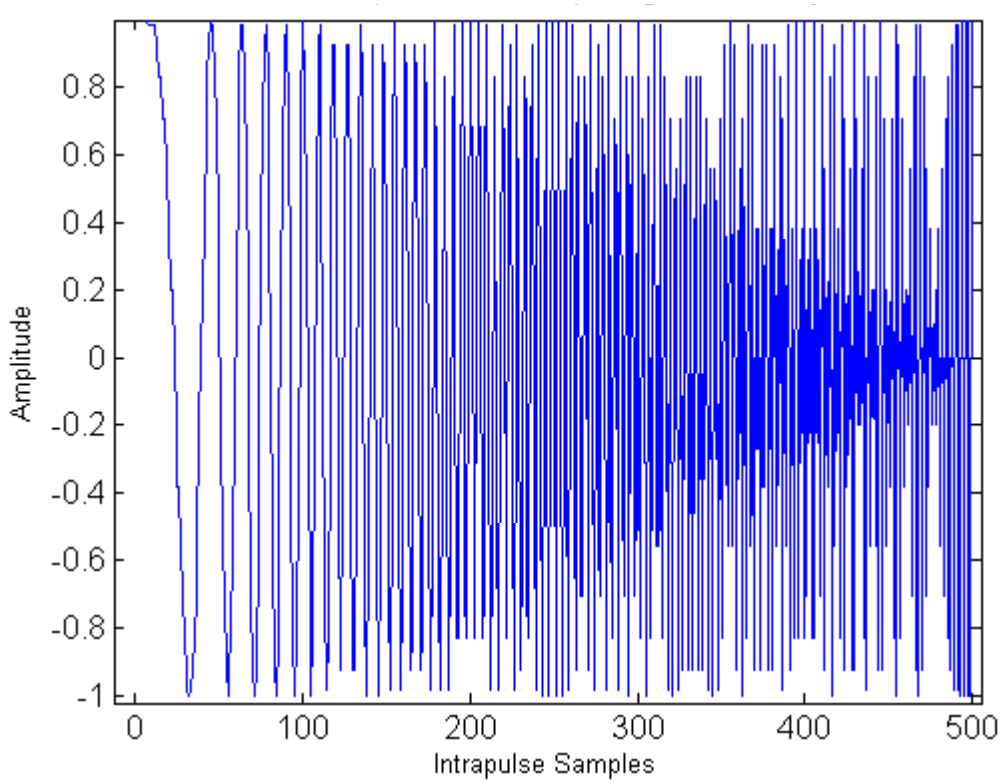


Figure 4.1 Intercepted ISAR Chirp as Generated by DRFM.

Figure 4.2 shows the range-Doppler profile of the 32-range bin target that is processed by the DIS simulation. The phase and magnitude coefficients will be extracted from this image and applied to the above chirp-pulse to create the normalized output waveform.

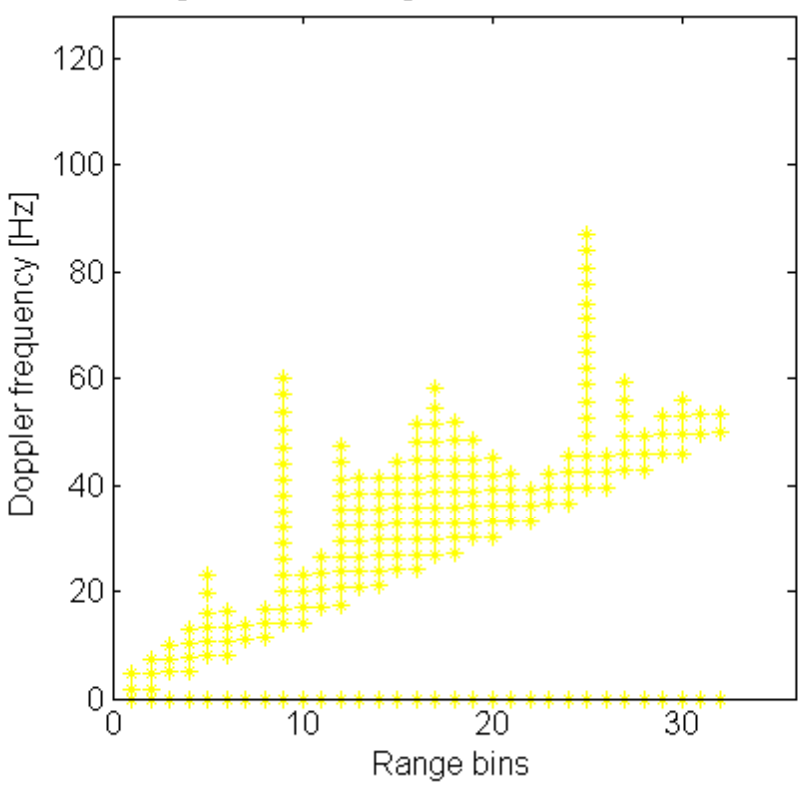


Figure 4.2 32 Range bin False Target Profile – 128 Pulses.

For the 32-range bin test case, the original software parameters are retained. The DIS integrates over 128 coherent pulses to form the output image. The extracted gain coefficients are quantized into 11 levels as shown in Figure 4.3. During the final summation, no overflows are recorded and the system produces the waveform shown in Figure 4.4.

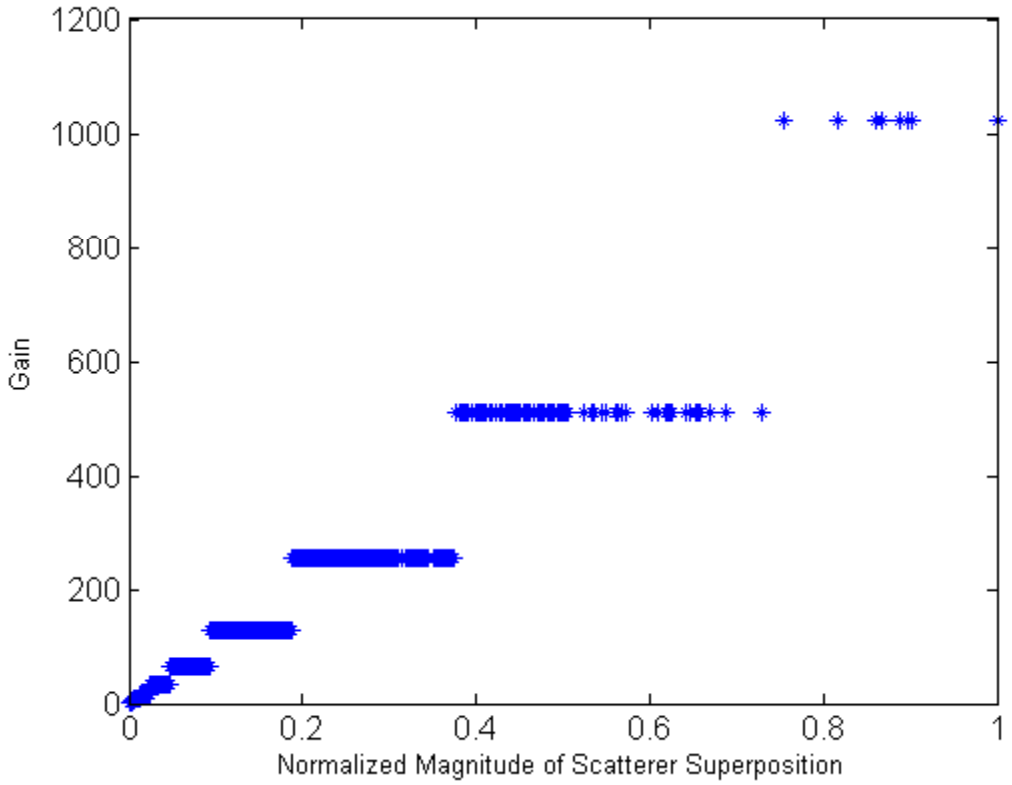


Figure 4.3 Gain Quantization Scheme for 32 Range bins.

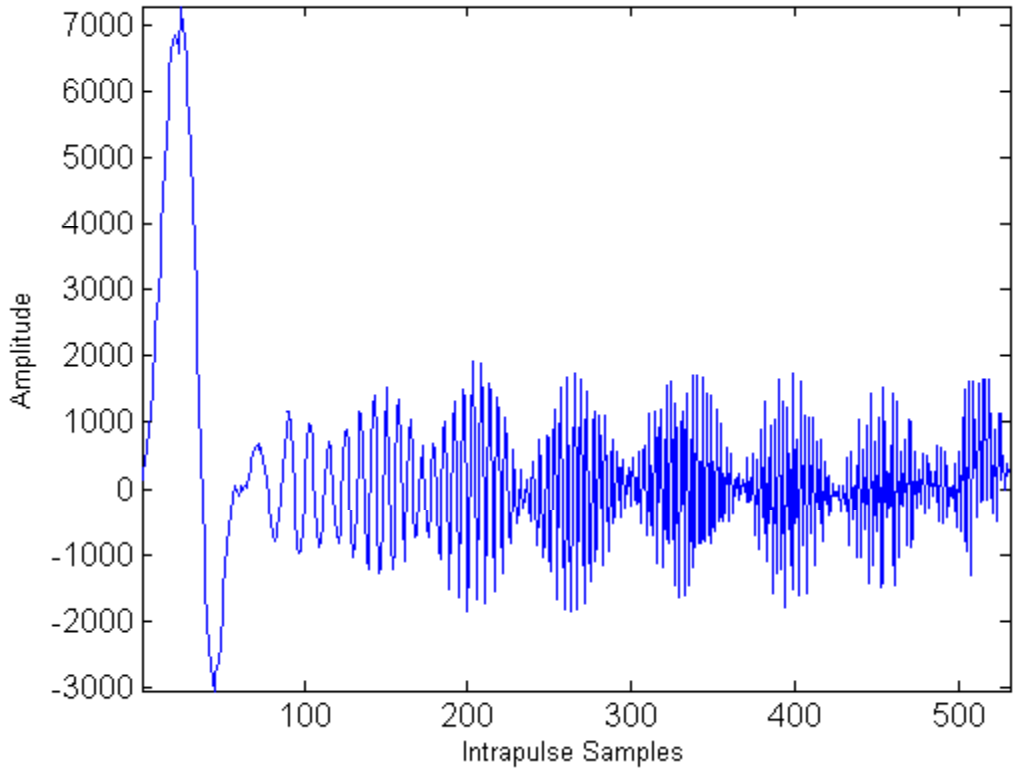


Figure 4.4 DIS Output Chirp Pulse for 32 Range bins.

The resulting output chirp-pulse shows a neatly modulated waveform as expected. However, keep in mind that this waveform only represents a target extent of 32 range bins. When the data is once again sampled and processed into a 2D range-Doppler image, as shown in Figure 4.5, it is very evident that the resolution is not high enough to depict an accurate ISAR image that is capable of fooling an AN/APS-137 operator. A 512-range bin target shows much higher resolution and image detail, provided the adder overflow is kept in check.

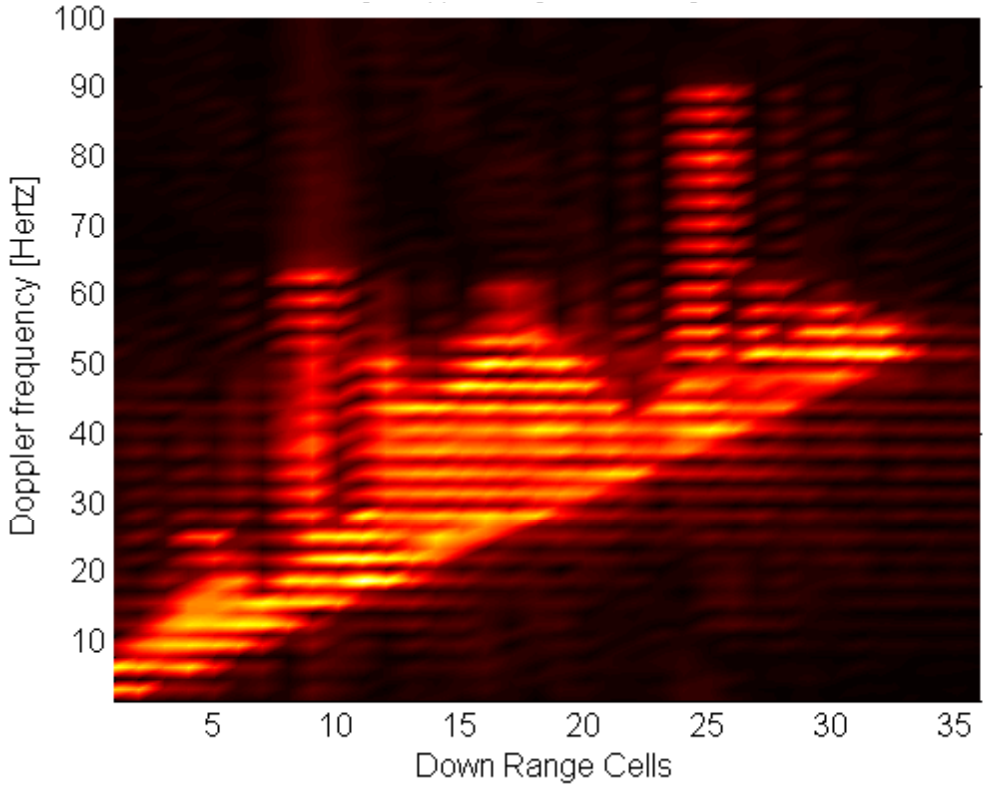


Figure 4.5 Range-Doppler Image for 32 Range bins.

2. 512 Range bins

The same simulation was run for an identical target profile, yet with a range resolution of 512 bins. Figures 4.6 and 4.7 show, respectively, the corresponding range-Doppler profile as well as the quantized magnitude coefficients as extracted from this profile.

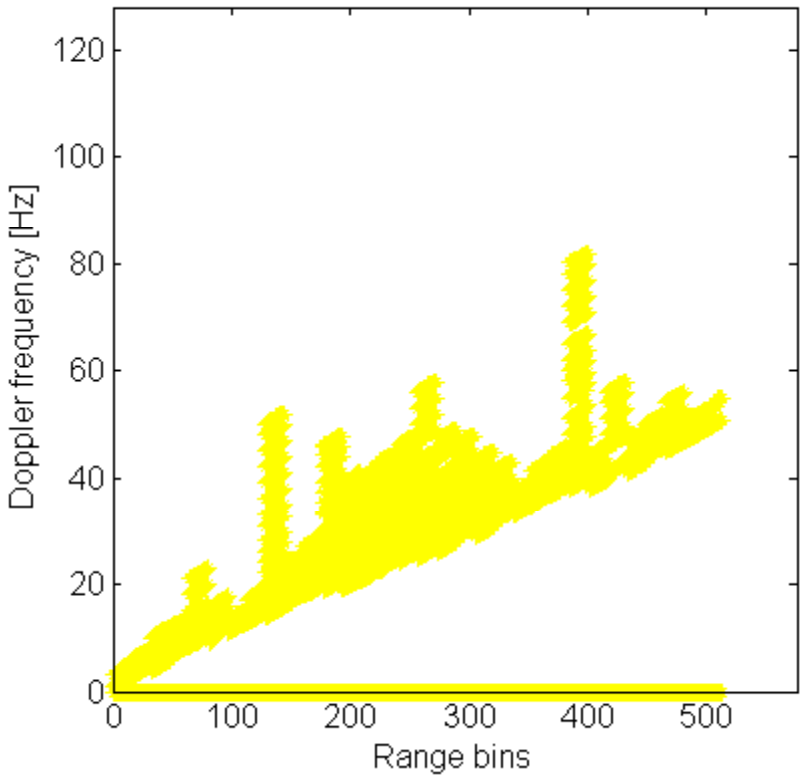


Figure 4.6 512 Range bin False Target Profile – 128 Pulses.

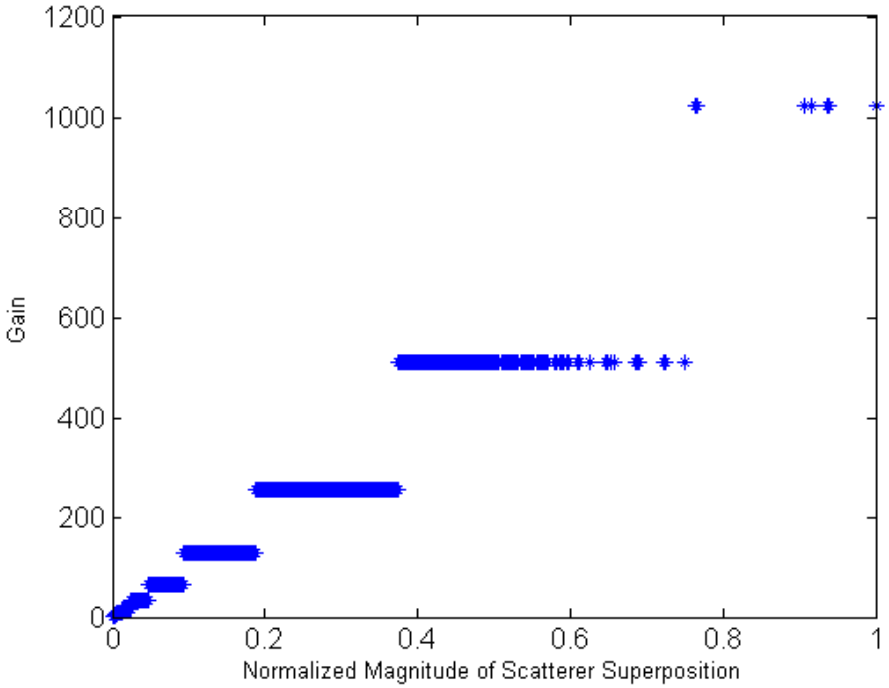


Figure 4.7 Gain Quantization Scheme for 512 Range bins.

Once again, the full 11-bit gain quantization scale is used as in the 32-range bin case. For all of the simulations, the same intercepted chirp-pulse as shown in Figure 4.1 is used as well. This time, however, the output waveform from the DIS is noticeably distorted as shown in Figure 4.9. Figure 4.8 shows the output waveform for the infinite-resolution case as a comparison.

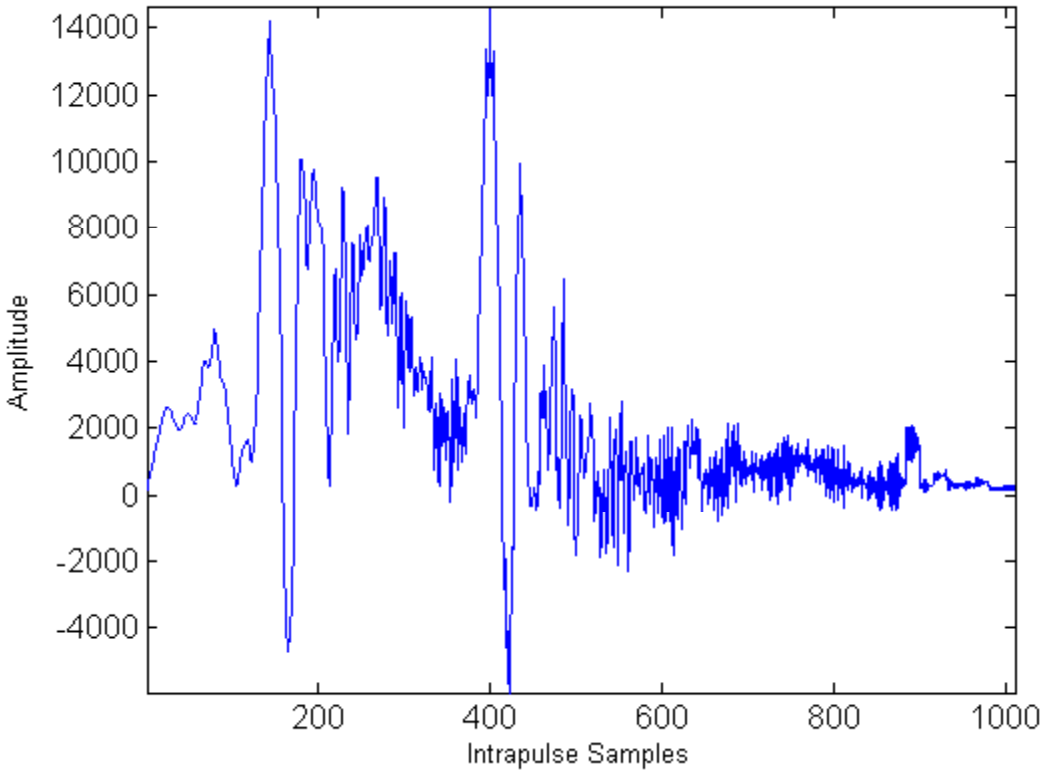


Figure 4.8 DIS Output Chirp Pulse for 512 Range bins, Infinite Resolution.

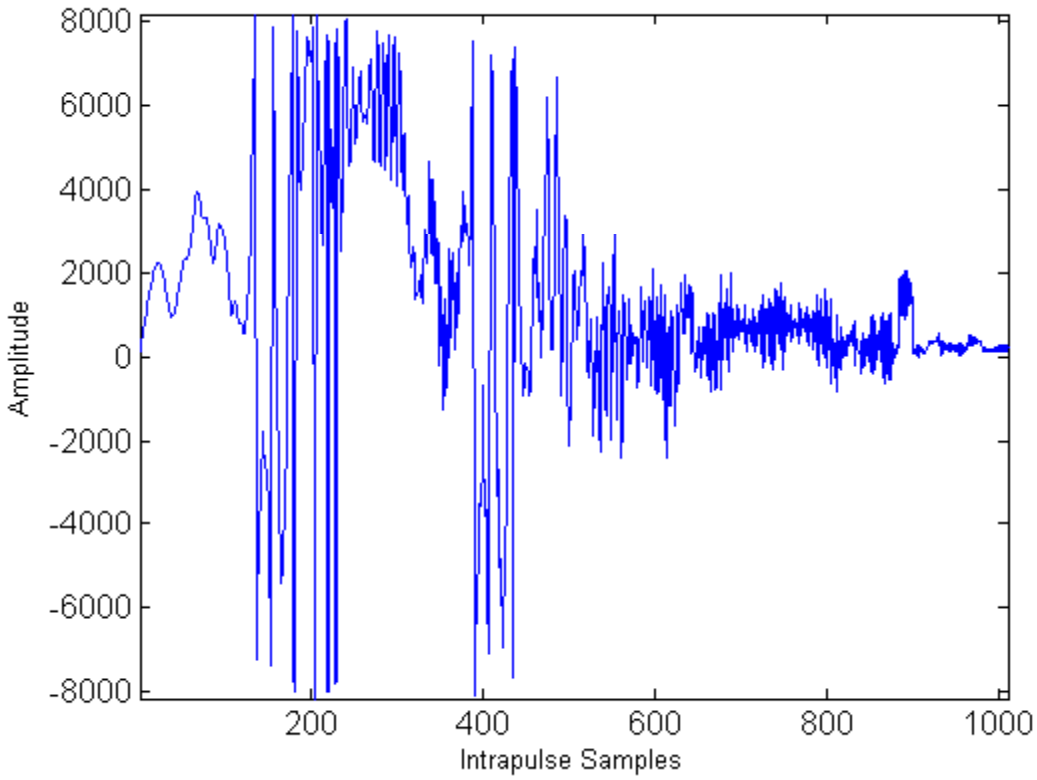


Figure 4.9 DIS Output Chirp Pulse for 512 Range bins, Uncapped.

Statistical measurements taken during the simulation show that an overflow in any one of the range bins occurred 25,774 times. Although the adder can only handle a maximum value of –8192 to 8191.75, a peak absolute value of 9,936 is recorded. Average noise within the waveform is calculated to be 0.0036. Moreover, the resulting 2D range-Doppler image, shown in Figure 4.11, is calculated to have a correlation coefficient of 0.8958 between the current image and the infinite-resolution image, shown in Figure 4.10. Nearly 11% of the resolution is lost simply due to overflow in the final summation adder of the DIS.

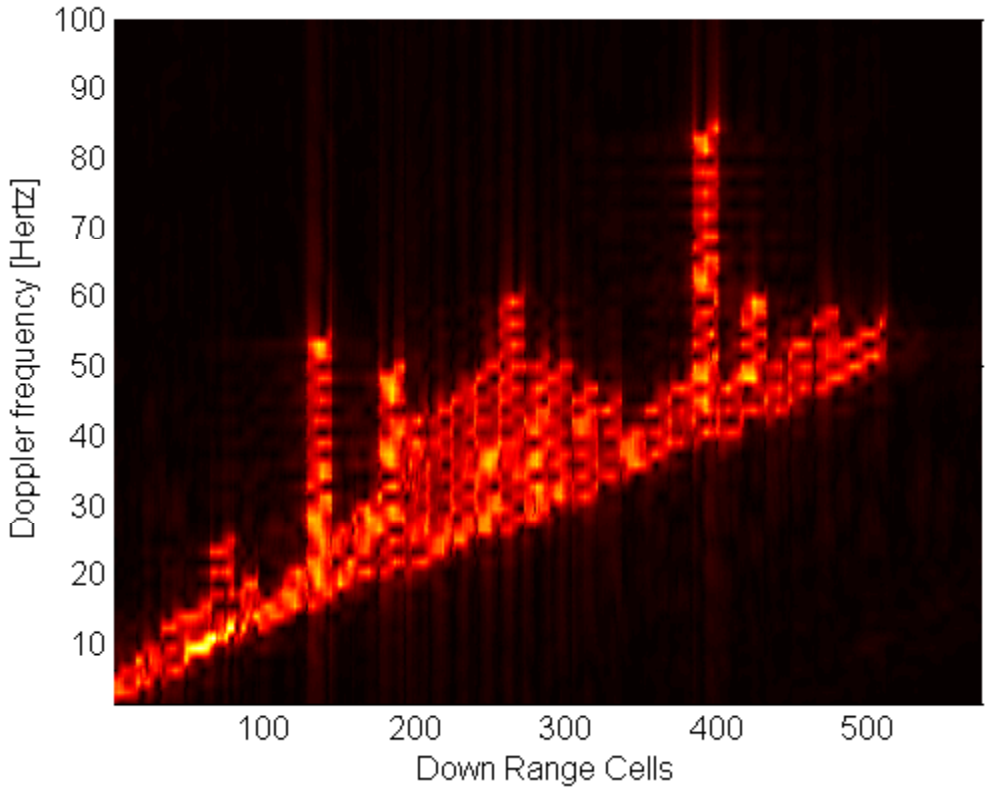


Figure 4.10 2D Range-Doppler Image for 512 Range bins, Infinite Resolution.

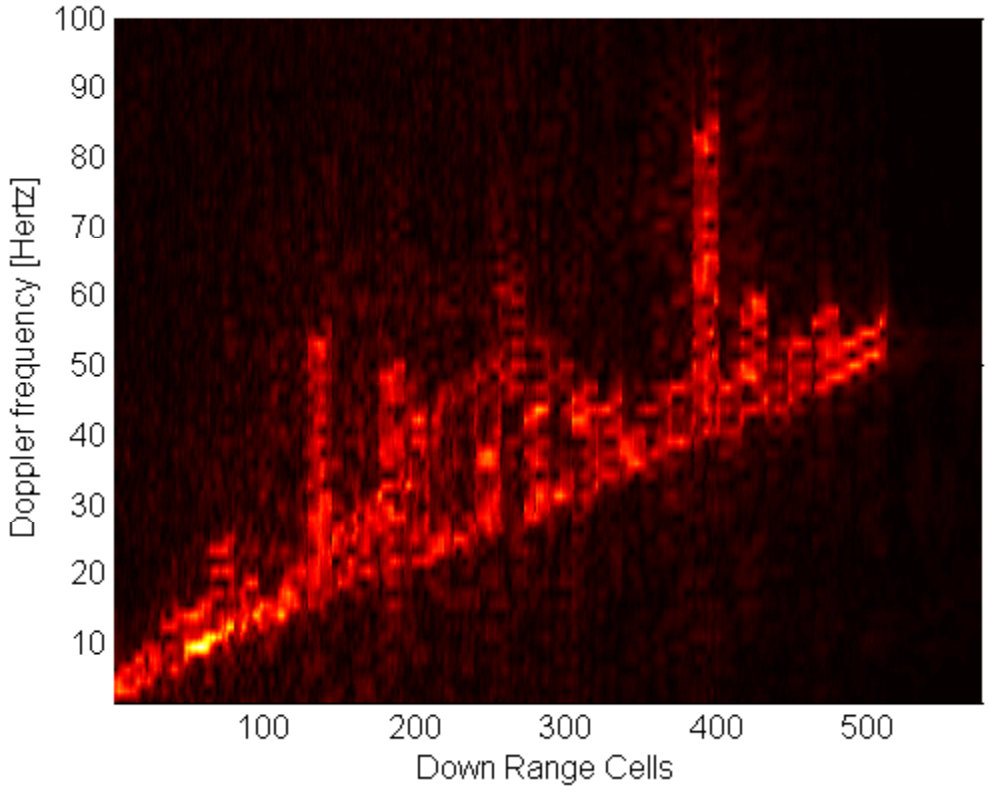


Figure 4.11 2D Range-Doppler Image for 512 Range bins, Uncapped.

D. DEALING WITH OVERFLOW

Three methods are examined to deal with the overflow. When examining them, the extent of image degradation is weighed against the cost of both the possibility of system redesign and the required DIS processing speed. The first method is stated simply to be complete, but is not recommended as a proper solution as it simply detects the problem and ignores it.

1. Signal Overflows with a Flag

Whenever an overflow exists, it can be very simply detected by the carry-out of the last stage of each addition. Upon detecting an overflow, the system can set a hardware flag and notify the user of the event at which point it is up to the operator to determine the extent of signal degradation and if the system must be stopped. Beyond not dealing with the issue, this approach allows for a system to be deployed with such a characteristic that can be exploited and used against the DIS to, in effect, invalidate all efforts at electronic deception.

2. Cap Overflows at a Maximum Value

The second approach still sets a flag when an overflow occurs, but this flag does not indicate any problems that would mean exploitation of system weakness. Instead, when an overflow is detected within any individual summation for each range bin and each pulse, the value of the overflow is discarded. The sum is then set at the maximum value allowed, either negative 8192 or positive 8191.75. This prevents the overflow from influencing the sign-bit of the adder, thereby placing a ceiling on image resolution. Figure 4.12 shows the output waveform for this approach within the MATLAB simulation. Figure 4.13 shows the resulting image produced from that waveform.

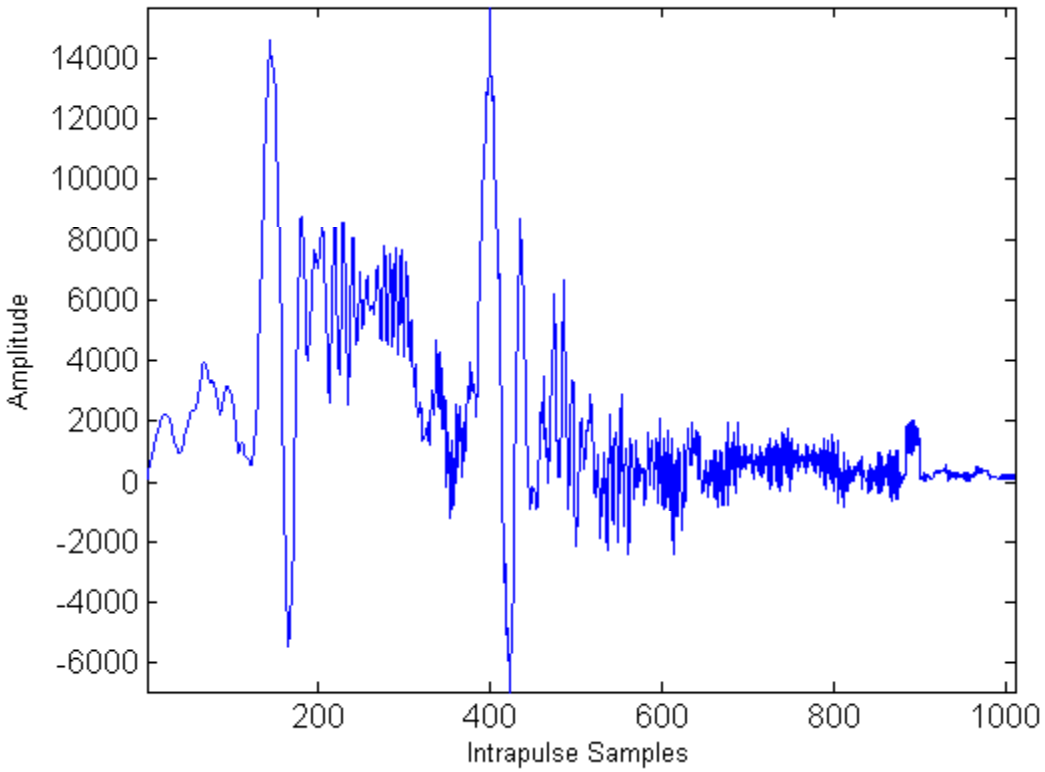


Figure 4.12 DIS Output Chirp Pulse for 512 Range bins, Capped.

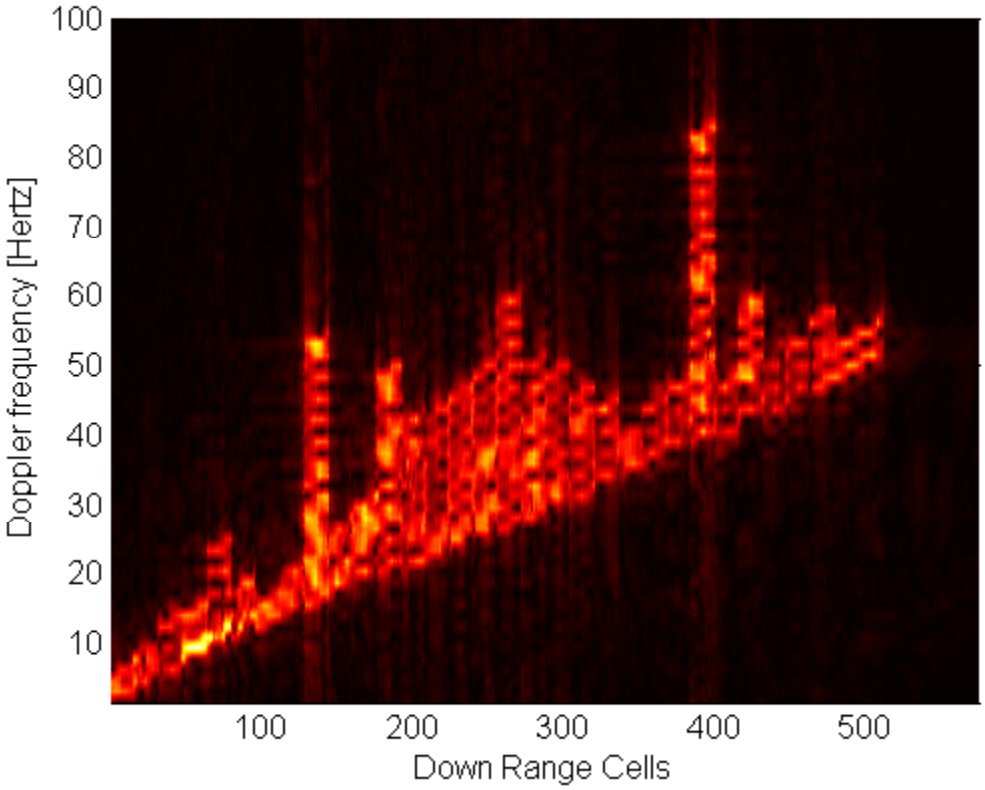


Figure 4.13 2D Range-Doppler Image for 512 Range bins, Capped.

The average noise coefficient is calculated to be 0.00034238 with an image correlation factor of 0.9898. After comparing the original overflow waveform with the capped case, it might be confusing to see that the amplitude is actually greater for the capped case. Moreover it might seem that the amplitude reaches above the previously stated summation limit of 8192. However, the amplitude shown is representative of the signal after it has been modulated and is not bound in amplitude by the range of the adder. Moreover, the uncapped waveform is generally smaller due to the fact that when an overflow occurs, the signal does not achieve a maximum value as in the capped version, but instead wraps around to a negative value. This is evident from the oscillation in the uncapped waveform around the 200 samples mark. Table 4.2 compares the results of the uncapped and capped simulations.

Parameters	Uncapped 512 Range bins	Capped 512 Range bins
Overflows	25774	Not Allowed
Peak Absolute Value	9936	8192
Noise Average	0.0036	0.00034238
Correlation Coefficient	0.8958	0.9898

Table 4.2 512 Range bin Statistical Comparison, Uncapped versus Capped.

Although capping each addition at a maximum value does prevent the signal degradation shown earlier, it comes at a fairly high cost. Extra hardware must be inserted after each stage of the adder, not only to detect the overflow, but also to reset the value of the sum to the maximum allowed positive or negative value. Implementing this approach sets back the design process significantly and must be avoided if at all possible.

3. Reduce the Gain

The third approach allows the overflow to proceed as in the original case. This time, however, the detection of an overflow is sent to the DIS controller. The DIS

controller receives the error input and, in response, lowers the bit-resolution of the gain quantization matrix by one bit. If an overflow continues, the controller may lower the quantization matrix by another bit.

Immediately there is an opportunity to improve the process. The DIS is run in real-time based upon stored coefficients. Whether or not an overflow occurs depends upon the coefficients. Therefore, the coefficients for each ship type may be analyzed ahead of time to determine the proper level of bit-quantization. As shown in the following figures, it is not always necessary to remove all overflows to obtain a reasonable range-Doppler image resolution that will fool an ISAR equally as well as the infinite-resolution image. There is thus a tradeoff between dynamic resolution and overall image quality that is best determined at the time of coefficient extraction. Although a predetermined quantization level should be sufficient to realize the DIS without noticeable image distortion, allowing the system controller to also dynamically modify quantization levels offers much greater flexibility on the battlefield as the DIS can adapt to the radar it aims to confuse.

a. 10-Bit Gain Quantization

Ten-bit gain quantization means that the power-of-two binary gain shifter is allowed to multiply the input pulses by a factor in the range from 2^0 to 2^9, or 1 to 512. This cuts the effective dynamic resolution in half; however it does not discard the top end of the gain spectrum. Instead, all of the coefficients are still normalized, but to within 10 bits of resolution instead of 11. Figure 4.14 shows the gain-quantization map for the same 512-range bin false-target profile using 10 levels.

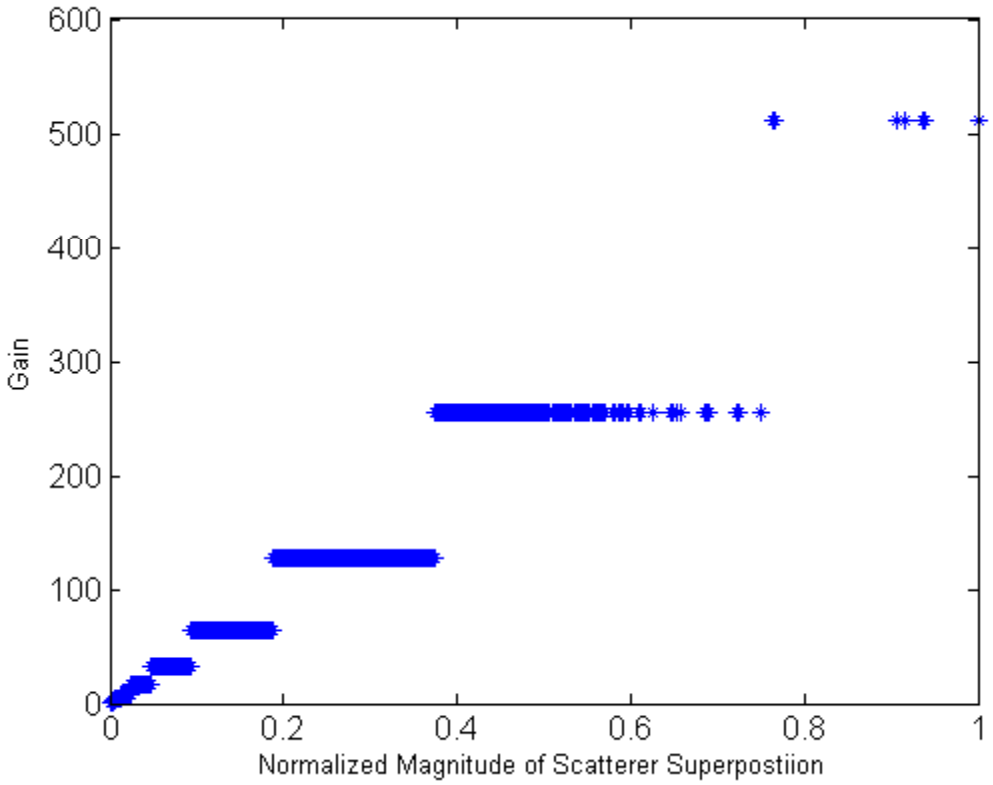


Figure 4.14 Gain Quantization Scheme for 512 Range bins, 10 Bits.

The output signal is shown in Figure 4.15, followed by the 2D range-Doppler image generated from the data. Notice how the maximum amplitude of the output waveform is half that of the capped version. According to simulation statistics, there are still 51 overflows with a peak value of 8373. However, the noise average and image correlation values are 0.00034432 and 0.9898 respectively. Although slightly more noise is present from lowering the gain, the image correlation is equal to that of the capped version to within 4 decimal places. The output range-Doppler images also appear visually indistinguishable.

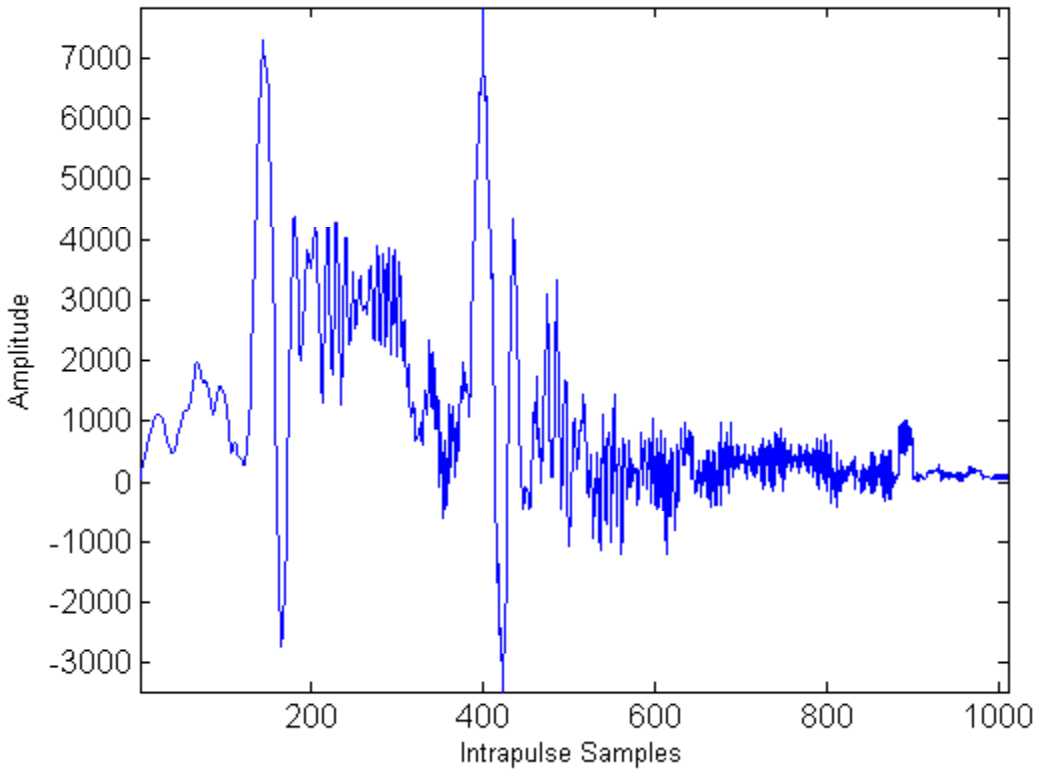


Figure 4.15 DIS Output Chirp Pulse for 512 range bins, 10 Bits.

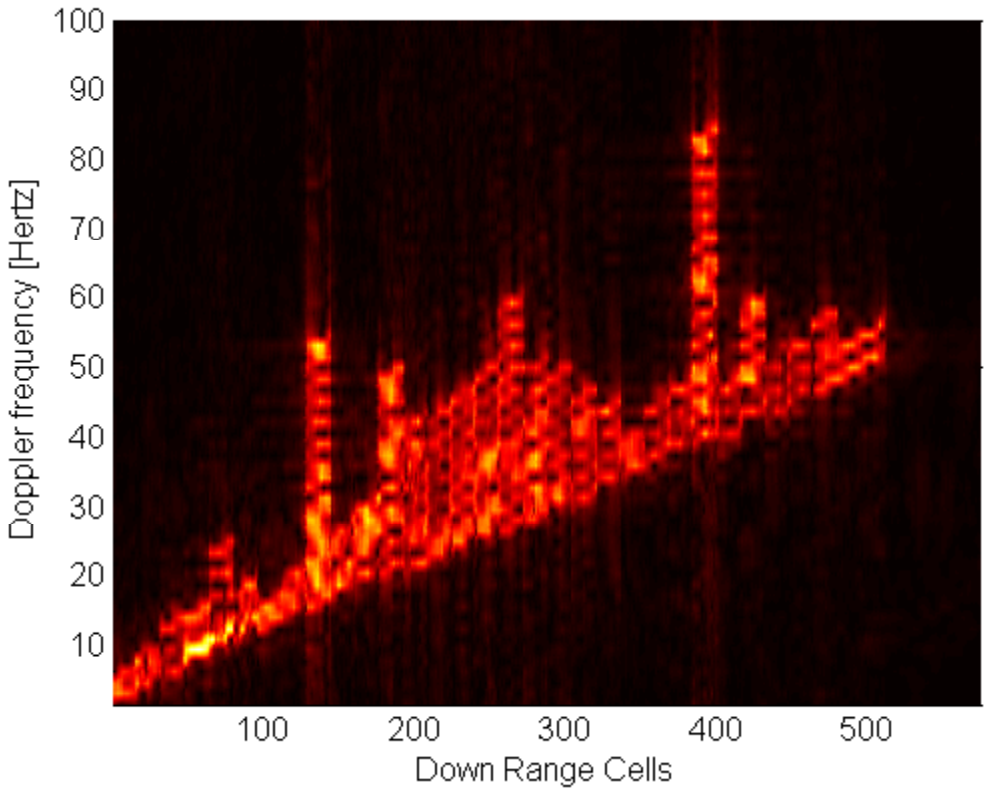


Figure 4.16 2D Range-Doppler Image for 512 Range bins, 10 Bits.

b. 9 Bit Gain Quantization

The experiment is taken one step further to explore the benefits of lowering the gain by one more bit to 9 total in an effort to remove all overflows. With 9 gain-quantization bits, modulated signals can be shifted to achieve multiplication in the range from 2^0 to 2^8 or 1 to 256. Once again, the dynamic range of the overall image is reduced by one half with regard to the 10-bit quantization scheme. Figure 4.17 shows the gain quantization map for the 9-bit case. In Figure 4.18 the amplitude of the output waveform is one-fourth of the magnitude of both the capped and infinite resolution versions.

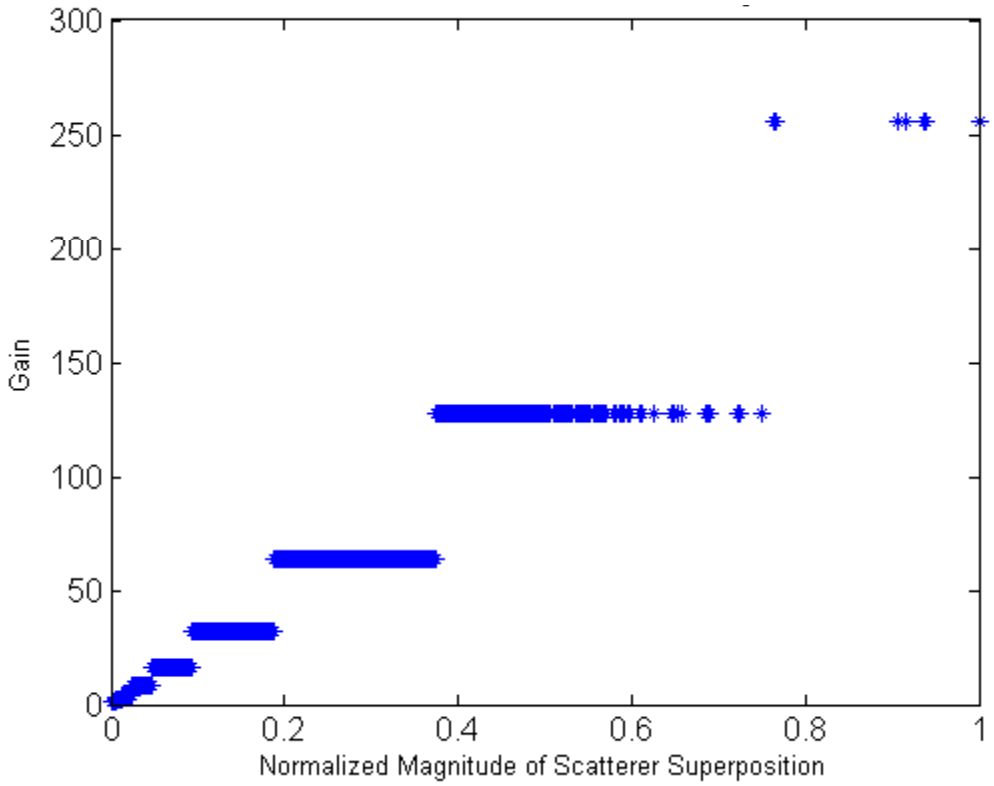


Figure 4.17 Gain Quantization Scheme for 512 Range bins, 9 Bits.

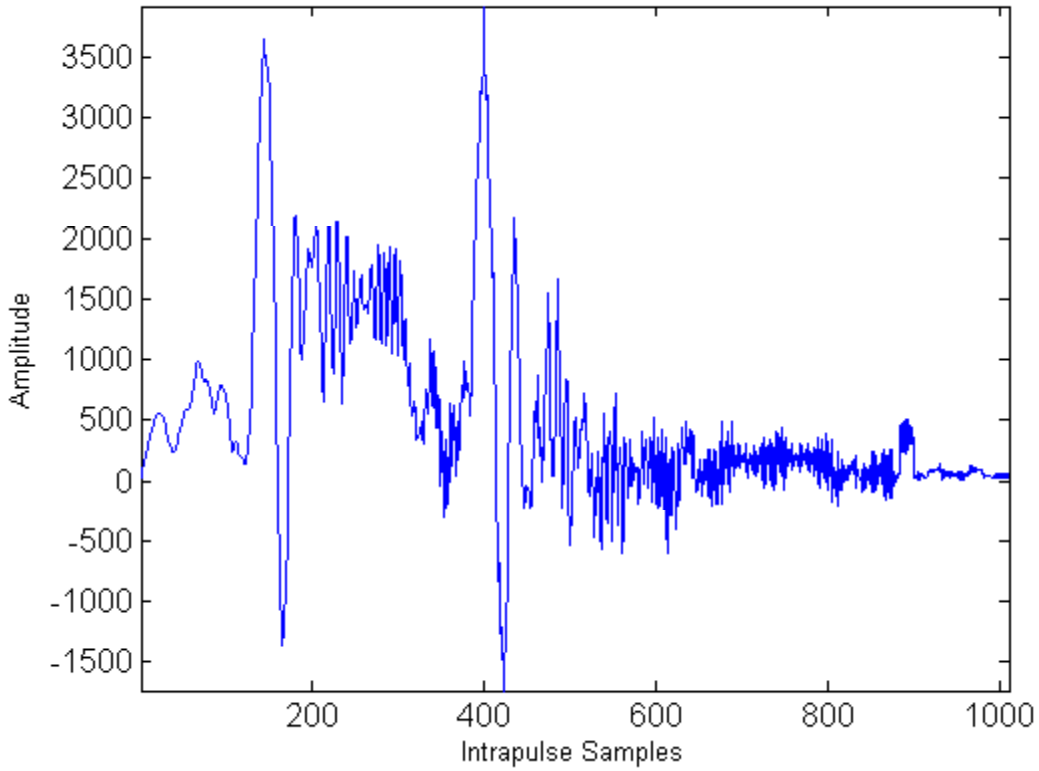


Figure 4.18 DIS Output Chirp Pulse for 512 Range bins, 9 Bits.

As expected, no overflows are recorded during the simulation. In fact, the peak summation value only reaches 4332, or roughly one-half of what is allowed. The resulting range-Doppler image is shown in Figure 4.19. The noise average is 0.00034315 with a correlation coefficient of 0.9898. Table 4.3 compares the results of the 10-bit and 9-bit quantization schemes.

Parameters	512 Range bins, 10 Bit	512 Range bins, 9 Bit
Overflows	51	0
Peak Absolute Value	8373	4332
Noise Average	0.00034432	0.00034315
Correlation Coefficient	0.9898	0.9898

Table 4.3 512 Range bin Statistical Comparison, 10-Bit versus 9-Bit.

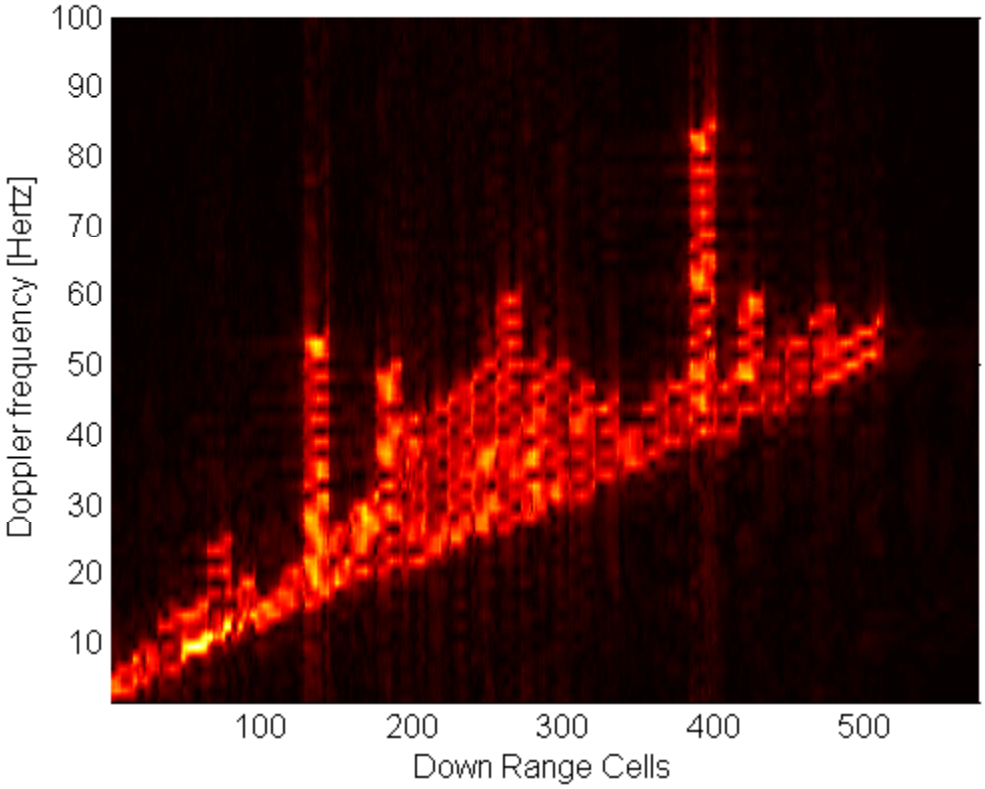


Figure 4.19 2D Range-Doppler Image for 512 Range bins, 9Bits.

E. SUMMARY

Based upon the current DIS design, generating false ISAR images for targets with a resolution of 512 range bins or greater presents a significant possibility of having an overflow occur in the final 16-bit summation adder. An overflow generally causes the sign bit to be affected, thus negating the result and thereby harming the overall signal quality and, ultimately, the final image quality. Such issues can be exploited by an enemy radar and thus pose a serious threat to the proper functioning of the DIS as a whole. Two non-trivial solutions exist for this problem.

The first solution is to detect when an overflow occurs in each range-bin summation, discard the overflow, and set the sum to the maximum allowed positive or negative value of the current adder stage. This method works well in improving both signal and image quality. However, it requires a very costly system redesign to implement and is thus not a viable option.

The second solution prevents overflows by reducing the bit-wise precision of the quantization of magnitude coefficients. By normalizing the coefficients to within 10 bits

of precision as opposed to 11 bits, overflows are significantly reduced and overall signal and image quality is nearly equal to the infinite-resolution case. Reducing the gain-quantization precision to 9 bits eliminates all overflows for the sample target profile; however it significantly reduces the dynamic range of the target as well. Therefore, the 10-bit solution is preferred as it allows for the current DIS design with minimal changes.

Lowering the gain modulation also serves to lower the total gain of the output signal and therefore also lowers the output power of the DIS. As a solution, the signal power may be adjusted during the up-conversion stage to a proper level before being retransmitted. Through careful analysis of the stored coefficients that run the DIS, the correct bit-quantization level can be determined ahead of time and with certainty to prevent overflow problems during battlefield operation.

Figures 4.20, 4.21, and 4.22 provide a graphical comparison of all methods discussed in terms of adder peak value, average noise, and correlation with the infinite-resolution image. The 16-bit summation adder overflow analyzed in this chapter, if left unchecked, does present a significant problem. However, these figures show that an expensive system redesign is not necessary as a more viable solution, which maintains image quality, is available through the adaptive adjustment of signal gain.

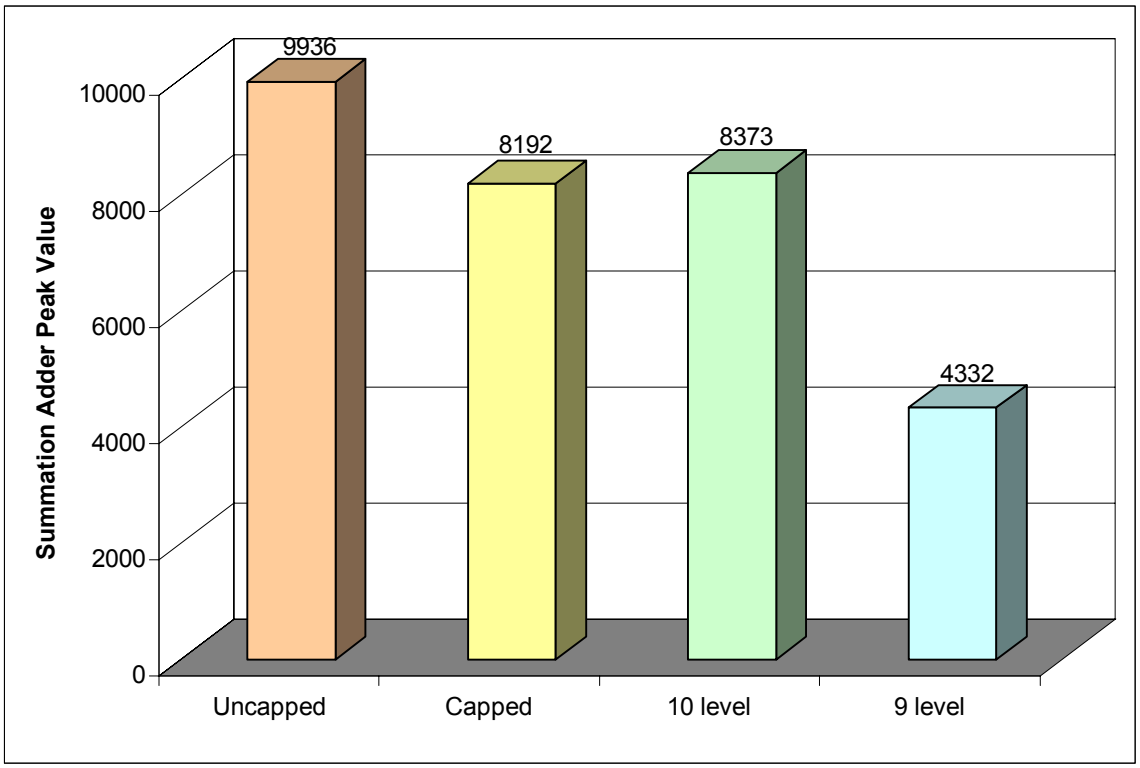


Figure 4.20 Peak Adder Value Comparison of all 512 Range bin Approaches.

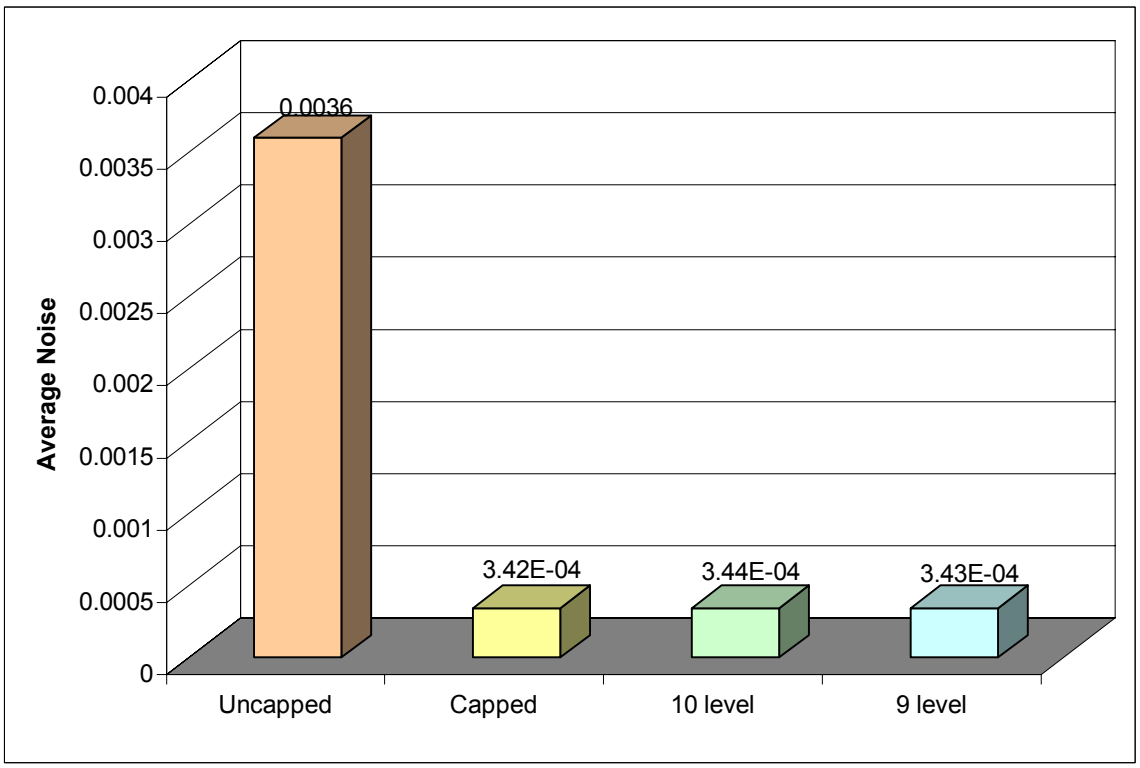


Figure 4.21 Average Noise Comparison of all 512 Range bin Approaches.

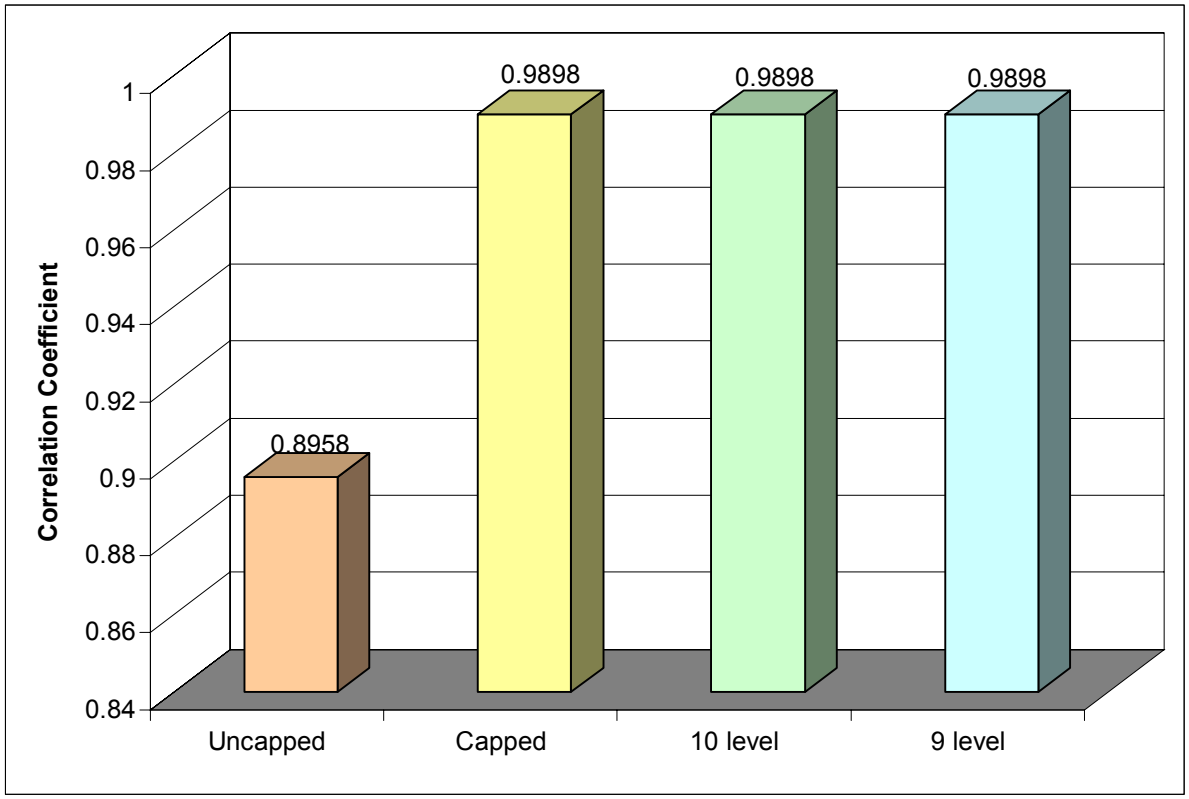


Figure 4.22 Correlation Coefficient Comparison of all 512 Range bin Approaches.

At this point the reader should have a proper understanding of both the DIS architecture and the image creation process. As mentioned above, the coefficients used to run the DIS are very important as they shape the entire set of output signal characteristics. As such, it is important to run the current MATLAB DIS simulation with a wider selection of realistic target coefficients. Chapter V examines this issue in more detail and provides an in-depth look at the coefficient extraction process.

THIS PAGE INTENTIONALLY LEFT BLANK

V. 2D ISAR IMAGE CREATION AND COEFFICIENT EXTRACTION

This chapter discusses the details behind forming a two-dimensional (2D) range-Doppler image from complex scattering elements received through chirp-pulse waveforms. Moreover, it is shown how to properly extract magnitude and phase coefficients from realistic simulation data. These coefficients are used to run the DIS and create false-target images that should be identical to the images originally created from the complex scattering elements. The resulting images form the basis for overlaying 2D range-Doppler profiles on a 3D target as discussed in Chapter VI. Through research into the 3D aspects of the 2D images, weaknesses within the current DIS algorithms are examined. Moreover, insight into practical implementations of 3D radar imaging is explored. The ultimate goal is to run both the DIS software-based and hardware-based implementations at acceptable speeds using coefficients from real-target data.

A. 2D RANGE-DOPPLER IMAGE CREATION

The general approach for forming a 2D ISAR range-Doppler image from received chirp-pulse return signals is discussed in Chapter II. Following the development in [6], the process begins with the receipt of return chirp-pulses of the form given in Equation 3.1. The return of each pulse represents individual complex scattering elements corresponding to a specific range and cross-range Doppler shift. N_p return pulses correspond to a target with N_r range bins. Range compression is performed by correlating the received signal with the reference function given by

$$s(m,n) = \text{rect}\left(\frac{t}{\tau}\right) e^{j(\pi\Delta/\tau)(m/f_s + nPRI)^2}, \tag{5.1}$$

whose transfer function is given by

$$S(k,n) = \frac{1}{N}\sum_{m=0}^{N-1} S(m,n) e^{-j\omega T_s km}. \tag{5.2}$$

The signal is transformed into the frequency domain by

$$I(k,n) = \frac{1}{N}\sum_{m=0}^{N-1} I(m,n)e^{-j\omega T_s km} \tag{5.3}$$

and then into complex range profile form by

$$C_r(m,n) = \frac{1}{N}\sum_{k=0}^{N-1} I(k,n)S^*(k,n)e^{-j\omega T_s km}. \tag{5.4}$$

Subsequent return pulses, aligned along coherent range bins, form a time history of range-profiles as in Figure 2.4. The rotational motion of the target over time, as in Figure 2.3b, creates a Doppler frequency shift in subsequent return pulses based on Equation 2.17 from which the cross-range resolution is given by Equation 2.18. Once in complex range-profile form, azimuth compression is completed based on

$$C(m,k) = \left| \frac{1}{N}\sum_{n=0}^{N_p-1} C_r(m,n)e^{-j\omega T_s kn} \right|, \tag{5.5}$$

the absolute value of the result of which is the desired 2D range-Doppler image as shown in Figure 2.1.

All realistic return data collected from an actual AN/APS-137 or similar simulation must proceed through this basic process in order to create the necessary 2D range-Doppler image. However each set of data, especially from simulations, has its own unique set of parameters used in the creation of the data, which must also be considered when performing the pulse-compression and conversion steps. Many of these parameters are actual AN/APS-137 tactical parameters and as such remain classified. The ability to simulate the DIS based on real sets of ISAR data with the classified parameters provides increased validity to the simulation.

In particular, some ISAR simulation data works in a sliding window fashion. At each single processing step, N_p return pulses are integrated. When the next pulse arrives, the window slides forward by one pulse to include the previous $N_p - 1$ pulses in addition to the current pulse. In this way, the created images represent a moving average of return data as opposed to a single slice in time. By averaging over all pulses in this

manner, the effects of single-pulse fluctuations are minimized, providing an overall higher quality image.

B. DIS COEFFICIENT EXTRACTION

The true test of the DIS lies in its ability to produce a false-target image of equal resolution and quality as an equivalent real-target image. The process by which the data is shaped into 2D range-Doppler images is described both above and in the previous chapters. This section will discuss the method by which magnitude and phase coefficients are extracted from the same data. Once extracted, these coefficients may be inserted in the DIS software model to verify correct false-target generation.

Following the development in [6], the target return signal is composed of individual scattering elements for each of R range bins according to Equation 3.11, reprinted below as

$$T(r,d,n) = A(r,d)e^{-j2\pi f(r,d)PRI}, \tag{5.8}$$

where $f(r,d)$ is the range-Doppler frequency shift of each scatterer and $A(r,d)$ is the corresponding scattering amplitude. For each range bin, r, the combined return signal is found by summing the complex scatterers along the azimuth dimension to achieve

$$T'(r,n) = \sum_{d=1}^{N_d(r)} T(r,d,n). \tag{5.9}$$

It is from this signal that the desired phase and gain coefficients are extracted for each range bin. The phase of each range bin-pulse combination is found by dividing the imaginary portion of Equation 5.9 by the real portion,

$$\phi_T(r,n) = \left\{ \frac{\Im(T'(r,n))}{\Re(T'(r,n))} \right\} \Big/ \frac{2\pi}{2^{k_p}}. \tag{5.10}$$

The phases are quantized to k_p-bit resolution to support the hardware implementation of the DIS. The phase from Equation 5.10 cannot be applied directly, however, as each target reflection must be represented as a phase increment on a pulse-to-pulse basis. The first pulse, $n = 0$, is given the value

$$\phi'_{inc}(r,0) = \phi_T(r,0). \tag{5.11}$$

For subsequent pulses, their phase is subtracted from the previous pulse and then added to the previous increment as

$$\phi'_{inc}(r,n) = \phi'_{inc}(r,n-1) + (\phi_T(r,n-1) - \phi_T(r,n)). \tag{5.12}$$

Integer values for the phase coefficients that will feed the DIS for each pulse and range-bin combination can then be obtained as

$$\phi_{inc}(r,n) = \lfloor \phi'_{inc}(r,n) \rfloor. \tag{5.13}$$

The gain coefficients are obtained from Equation 5.9 by taking the normalized magnitude of the combined return signal for each range bin and pulse combination as [1],

$$T'_N(r,n) = \frac{|T'(r,n)|}{\max|T'(r,n)|}. \tag{5.14}$$

This method normalizes all of the gain coefficients linearly to a value between 0 and 1. As discussed in Chapter IV, the gain is quantized into a maximum of 11 unique levels. Selection of the quantization scheme determines the dynamic range of the corresponding ISAR image. Proper selection is particularly important when considering the fact that the overall gain may need to be reduced to prevent an overflow in the final 16-bit summation adder as discussed in the previous chapter.

C. SUMMARY

The image processing steps discussed above are implemented in a corresponding MATLAB software model. This model is based on the work of Fernando Le Dantec [1] and follows the process outline given in Figure 5.1.

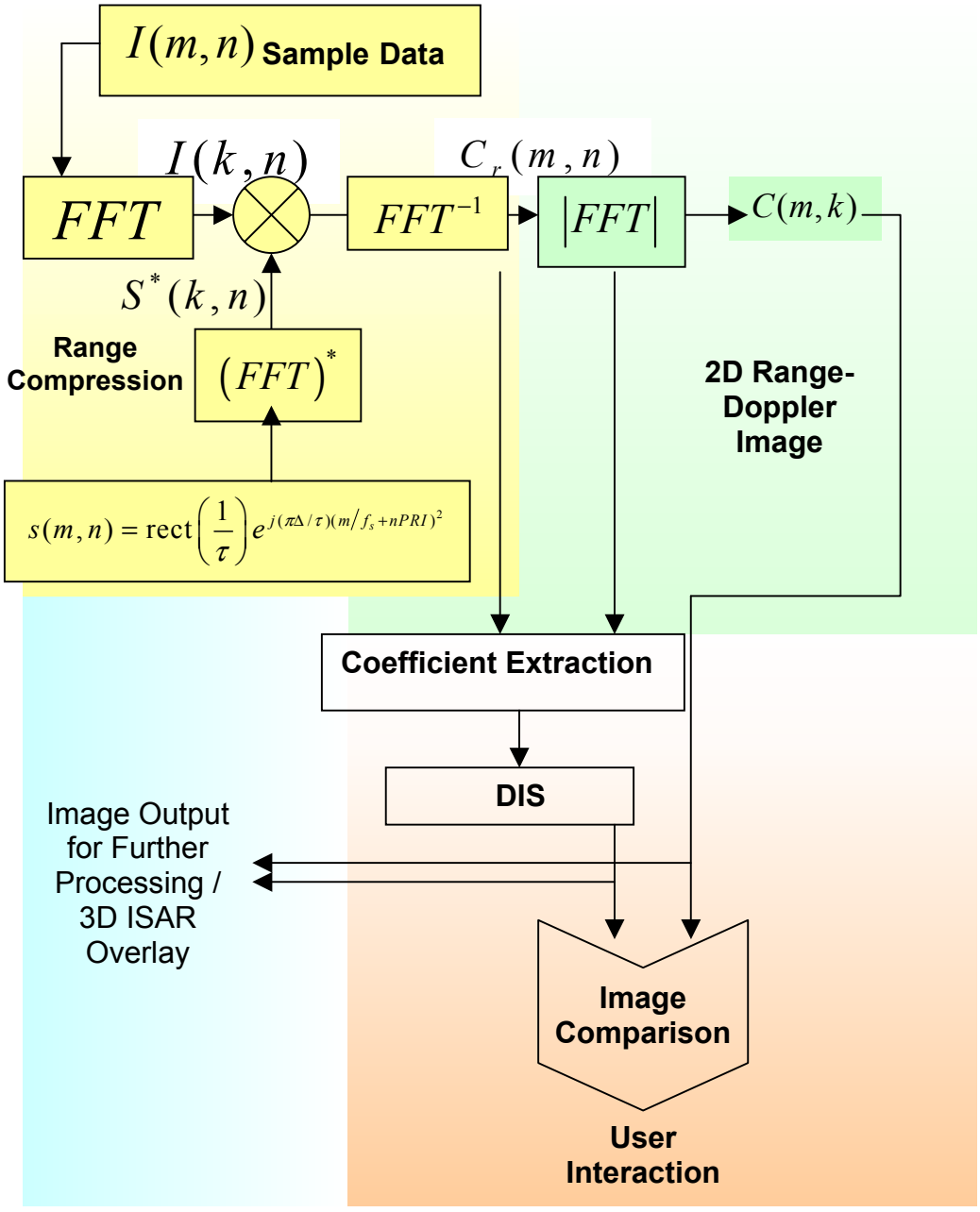


Figure 5.1 Image Processing Block Diagram of MATLAB Model.

The software was created using sample data and as such may need to be moderately tweaked when applied to the processing of real ISAR data. Specifically, depending on what form the date set is in, not all of the steps listed in Figure 5.1 may need to be completed. The software model assumes that the data represents the complex return signal as would be received by the AN/APS-137 radar during operation. It therefore requires range compression based on correlation with a reference function to create a complex range profile for each range bin. The 2D range-Doppler image is then created through azimuth compression based on Equation 5.7. Once the signal transformation is complete, gain and phase coefficients are extracted as previously described and subsequently run through the original DIS simulation from [1].

Care is taken to provide a variable that allows the correct sliding window size to be set. This ensures that correct moving-average signal integration is completed. Either the images created directly from ISAR compression or indirectly through the DIS simulation can be used to run the 3D software overlay model presented in Chapter VII. Before the detailed discussion of this model is given, however, Chapter VI presents significant advances already made into the field of 3D ISAR.

VI. THREE DIMENSIONAL (3D) ISAR

Two-dimensional (2D) ISAR images in range-Doppler format are widely used for long-range target detection and classification. Working within the range-azimuth medium, there are several inherent shortcomings associated with traditional ISAR. First, because the cross-range dimension is directly related to the rotation of the target, small rotational aspect angles present a challenge to the radar in determining the correct scale in the cross-range dimension without prior knowledge of target motion. Moreover, the cross-range dimension of ISAR has no real scale, and thus does not present an accurate physical description of the target. Finally, although a 2D ISAR image is a projection of a 3D target, traditional processing methods cannot resolve the third dimension of target altitude.

The benefits of being able to image a target at full range in three dimensions are several-fold. The ability to create a precise 3D image provides the capability to target specific features of a potential threat for laser-guided munitions. Within the littoral combat domain, harbors are filled with numerous ships, both of the commercial and warship variety. The combined motion of multiple targets within a compact region makes it difficult to pick out a specific ship at full range, especially when smaller, fast moving boats are more likely to present the threat. The AN/APS-147 is targeted specifically at this close-combat region and is thus an ideal candidate for 3D ISAR imaging. Through comparison of 2D images and their corresponding 3D counterparts, it becomes possible to further classify and distinguish the more subtle components of current 2D images and thus provides enhanced capability to existing combat radar systems.

This chapter presents a discussion of recent research attempts at synthesizing 3D ISAR images. It begins with an overview of three-channel monopulse radar, followed by a detailed discussion of 3D interferometric ISAR imaging through the use of two receiver antennas at differing altitudes. These methods are compared to a three-receiver ISAR system aimed at accurately solving for the cross-range scale of a target to be able to create three motion-compensated orthogonal radar views that, when combined, form a 3D

ISAR image. Two of the three approaches rely upon backscattering phase information contained in current 2D ISAR images. Therefore the chapter ends with a short discussion of modifications that might be made to the DIS to support 3D ISAR false-target imaging.

A. 3D MONOPULSE RADAR

The goal of three-channel monopulse radar is to resolve each range point cell in two orthogonal cross-range dimensions, specifically azimuth and elevation. This is accomplished by using either chirp-pulse or stepped-frequency waveforms composed of one sum and two difference channels. The difference channels represent differential error signals based on either azimuth versus range or elevation versus range calculations [2]. The basic diagram of the three-channel monopulse process is shown in Figure 6.1. The bandwidth of the radar signals is chosen to match the desired slant-range resolution of the system. Traditional processing methods using the Discreet Fourier Transform convert each burst of echo signals within each channel to a corresponding synthetic range profile. The error signals in each of the two difference channels are further normalized by range cell to produce bipolar cross-range position data. The resulting image is displayed by plotting both azimuth and elevation profiles against the slant-range dimension. Similar to traditional ISAR, Doppler frequency values are still obtained via the rotational motion of the target.

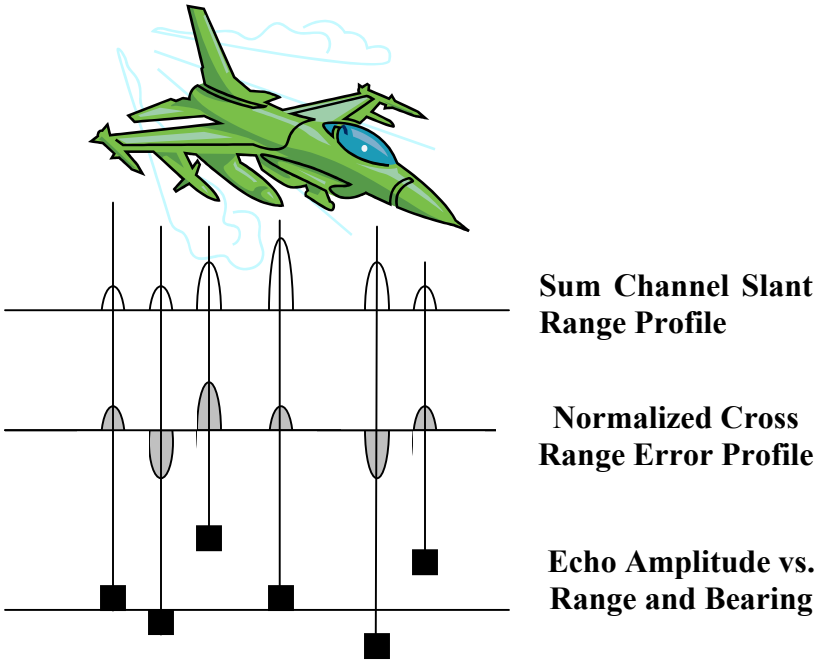


Figure 6.1 Monopulse Cross-Range Error Signals (From Ref. [2].)

An example of the 3D display setup for this approach is shown in Figure 6.2. This method has been studied at the Naval Research Laboratory. Two significant challenges posed include resolving multiple scattering centers within range cells as well as achieving high-resolution results at long range. Early research has shown that the cross-range resolution decreases with increased target range. The reason is that the difference-channel signals are based on the cross-range angular positions of resolved target reflections and become smaller with increased range as signal voltage approaches the receiver noise level. Achieving the maximum range possible typically requires use of stepped-frequency waveforms as they already include inherent pulse-to-pulse coherent integration. In general, widespread research of 3D monopulse radar is limited as increasing attention is focused on more recent techniques including interferometric imaging [2].

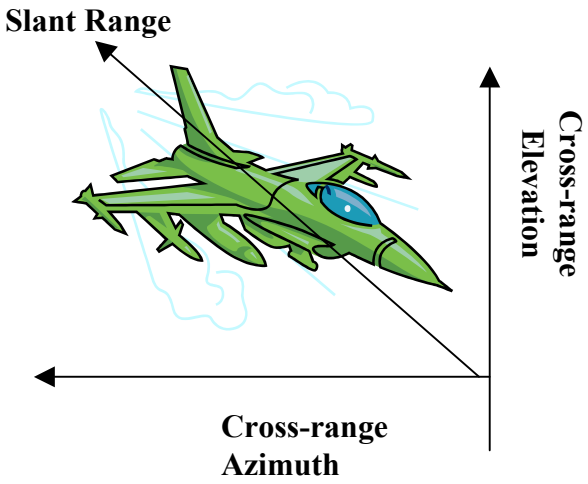


Figure 6.2 3D Monopulse Radar Image (From Ref. [2].)

B. 3D INTERFEROMETRIC ISAR

Interferometric imaging has long been in use for 3D terrain mapping using SAR. It reconstructs elevation based on imaging a given section of land at two different angles and correlating the resulting image. 3D interferometric ISAR relies on the same general approach and aims to correlate the return signals of two 2D ISAR images obtained at different altitudes to produce the third dimension. This extra elevation data is extracted from the relative phase shift, or back scattering effects, between adjacent range cells of each 2D image.

Three significant issues are present when considering the 3D interferometric ISAR approach. First, uncontrolled phase scintillations, or target glint, make it difficult to correlate neighboring range cells across both images. This problem is however lessened through the use of high-resolution wideband radar and is thus much less an issue with modern radar systems.

Secondly, typical ISAR applications include imaging small targets, such as aircraft or ships, as opposed to wide stretches of land in the traditional SAR mode. This forces the radar system to use more dedicated data processing power to extract the desired target from the surrounding region. Once again, simply using a high enough resolution radar system can overcome the effects of this issue.

Thirdly, and most importantly, 3D interferometric imaging requires that the phases of corresponding range bins in each image be "unwrapped." Phase values can only be uniquely determined in a range of 360 degrees. Typical phase measurements are made from $-\pi$ to π. However, actual phase values, which also represent velocity, are not restricted to this range. Values outside of this range are aliased back in, or basically "wrapped" around. The challenge is to properly extract the entire phase shift from the received data. This problem has been studied for over twenty years and yet is still the single largest stumbling block for interferometric processing.

In order to completely unwrap a phase, it must both be in the range from $-\pi$ to π and the magnitude of the complex image at all points must be nonzero. Through increased sampling, the first condition can be met. However, unlike land-based terrain, typical ISAR images are composed of localized scattering centers, leading to discontinuities. For testing purposes, this issue can be overcome by considering the target being imaged to be appropriately small enough that proper antenna placement assures reasonable signal returns based upon the Nyquist sampling theorem. When scaled up to the size of an aircraft carrier, however, this issue needs considerably more attention to obtain valid results.

Following the development in [8], the geometry setup for a bistatic 3D interferometric ISAR is shown in Figure 6.3. The target rotates about an axis perpendicular to the radar line of sight. The two antennas are aligned such that their coordinates are

$$(-R_{r0}\sin(\beta_{tr}/2), -R_{r0}\cos(\beta_{tr}/2), h_1) \tag{6.1}$$

and

$$(-R_{r0}\sin(\beta_{tr}/2), -R_{r0}\cos(\beta_{tr}/2), h_2), \tag{6.2}$$

where

R_{r0} is the distance from the receiving antenna to the target origin,

β_{tr} is the angle between the base of the transmit and receive antennas,

h_1 is the height of the first receiving antenna, also transmitting, and

h_2 is the height of the second receiving antenna.

Basically they are right above each other. The transmitting antenna is also a receiving antenna in this configuration.

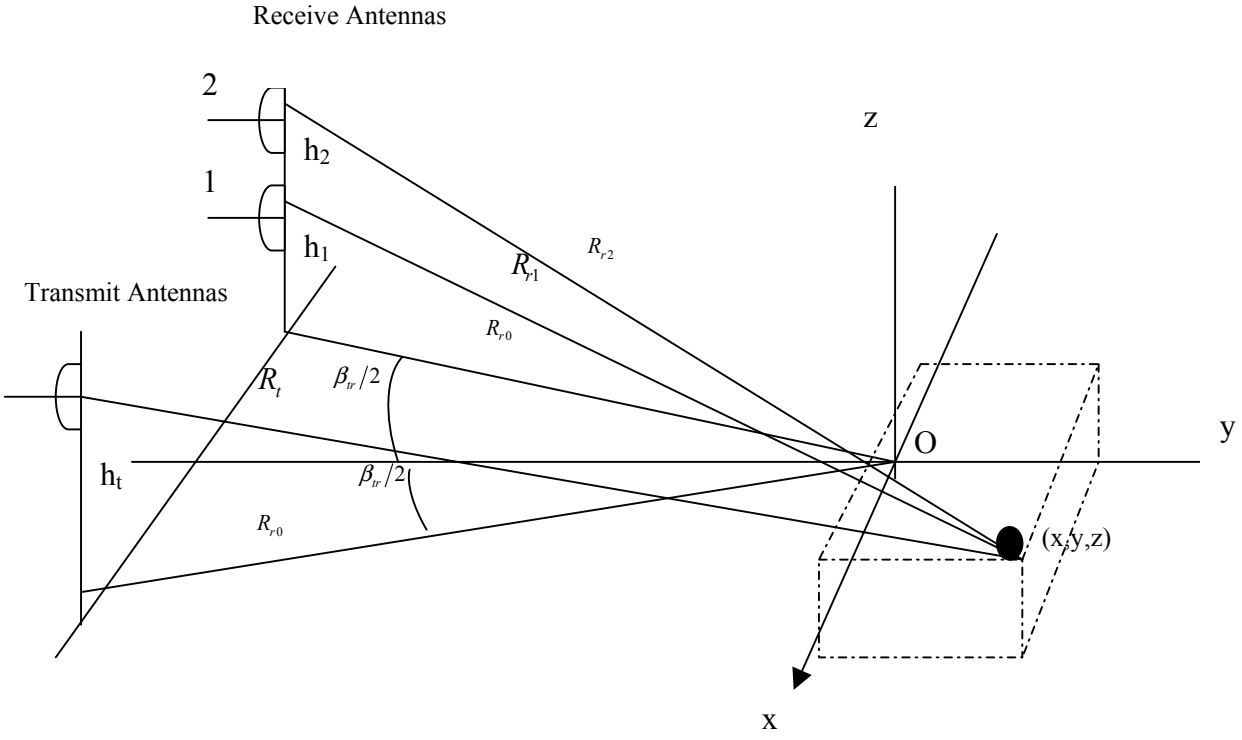


Figure 6.3 Interferometric ISAR Antenna Setup(After:[8]).

A general 3D target scattering density function, $f(x,y,z)$, is assumed and scatters can be measured over a rotating azimuth angle θ in the range $(-\theta_{\max}/2) \le \theta \le (+\theta_{\max}/2)$. Under this setup, the *i*-th scattering field can be measured by

$$E_{si}(f,\theta) = C_0 \iiint_{D^3} f(x,y,z) e^{-j\frac{2\pi}{\lambda}(R_t+R_{ri})} dxdydz, \qquad 6.3$$

$$R_t(x,y,z,\theta) = \sqrt{R_{t0}^2 + x^2 + y^2 + (z-h_t)^2 - 2R_{t0}\left[x\sin\left(\theta+\frac{\beta_{tr}}{2}\right) - y\cos\left(\theta+\frac{\beta_{tr}}{2}\right)\right]},$$

6.4

and

$$R_{ri}(x,y,z,\theta)=\sqrt{R_{r0}^2+x^2+y^2+(z-h_i)^2-2R_{r0}\left[x\sin\left(\theta-\frac{\beta_{tr}}{2}\right)-y\cos\left(\theta-\frac{\beta_{tr}}{2}\right)\right]},$$

6.5

where

C_0 is a complex constant,

D is the target extent, and

f is the frequency in the range $f_{\min}\le f\le f_{\max}$.

A 2D interferometric ISAR image can be computed by restricting the altitude to $z=0$ and solving the 2D version of Equation 6.3 as

$$E_s(f,\theta)=\iint_{D^2} f_z(x,y)e^{-j\frac{2\pi}{\lambda}[R_t(x,y,0,\theta)+R_{ri}(x,y,0,\theta)]}dxdy, \qquad 6.6$$

where $f_z(x,y)$ is an integrated version of $f(x,y,z)$. By solving for $f_z(x,y)$ in Equation 6.6, we have

$$f_z(x,y)=\int_0^{\infty}\int_0^{2\pi} E_s(f,\theta)e^{j\frac{2\pi}{\lambda}[R_t(x,y,0,\theta)+R_{ri}(x,y,0,\theta)]}df\,d\theta. \qquad 6.7$$

This represents a coherently focused 2D scattering function at a given z altitude plane. Applying the previously stated azimuth angle boundaries gives

$$\hat{f}_z(x,y)=\int_{f_{\min}}^{f_{\max}}\int_{-\theta_{\max}/2}^{+\theta_{\max}/2} E_s(f,\theta)e^{j\frac{2\pi}{\lambda}[R_t(x,y,0,\theta)+R_{ri}(x,y,0,\theta)]}df\,d\theta. \qquad 6.8$$

Traditional 2D ISAR FFT-based approaches can then be used to project the resulting image into the 2D plane.

The return of each antenna in the bistatic setup can thus generate a two-dimensional image in the format just described. It is important to note that during this whole process, the phase information at each point in the pixel array has not been altered.

In order to generate the proper continuously varying altitude information based on the phase shifts in each 2D image, it is necessary to accurately describe the phase difference between two images for a given pixel. The general expression for this phase difference is given as

$$\Delta\phi(x,y,\alpha_1,\alpha_2)=\frac{2\pi}{\lambda}[R_{r1}(x,y,z,0)-R_{r2}(x,y,z,0)] \tag{6.9}$$

where

R_{r1} is the distance between the target and first receiving antenna,

R_{r2} is the distance between the target and second receiving antenna, and

α_i is the depression angle of the *i*-th receiving antenna calculated as

$$\alpha_i=\tan^{-1}\frac{h_i}{R_{r0}}. \tag{6.10}$$

Through a Taylor expansion of R_{ri}, the phase difference can be expressed in the following equations:

$$\Delta\phi(x,y,z,\alpha_1,\alpha_2)=\frac{2\pi}{\lambda}[A(\alpha_1,\alpha_2)z^2+B(x,y,\alpha_1,\alpha_2)z+C(x,y,\alpha_1,\alpha_2)], \tag{6.11}$$

$$A(\alpha_1,\alpha_2)=\frac{1}{2R_{r0}}k_1(\alpha_1,\alpha_2), \tag{6.12}$$

$$B(x,y,\alpha_1,\alpha_2)=(\sin\alpha_2-\sin\alpha_1\left[1-\frac{x\sin\frac{\beta_{tr}}{2}-y\cos\frac{\beta_{tr}}{2}}{R_{r0}}k_2(\alpha_1,\alpha_2)\right], \tag{6.13}$$

$$\begin{aligned}C(x,y,\alpha_1,\alpha_2)=\Bigg[&\frac{R_{r0}}{\cos\alpha_1\cos\alpha_2}-x\sin\frac{\beta_{tr}}{2}-y\cos\frac{\beta_{tr}}{2}-\frac{x^2+y^2}{2R_{r0}}\\&+\frac{\left(x\sin\frac{\beta_{tr}}{2}+y\cos\frac{\beta_{tr}}{2}\right)^2}{2R_{r0}}k_3(\alpha_1,\alpha_2)\Bigg](\cos\alpha_2-\cos\alpha_1),\end{aligned} \tag{6.14}$$

$$k_1(\alpha_1,\alpha_2)=\cos^3\alpha_1-\cos^3\alpha_2, \tag{6.15}$$

$$k_2(\alpha_1,\alpha_2) = 1 - \sin^2\alpha_2 - \sin^2\alpha_1 - \sin\alpha_2 \sin\alpha_1, \tag{6.16}$$

and

$$k_3(\alpha_1,\alpha_2) = \cos^2\alpha_1 + \cos\alpha_1 \cos\alpha_2 + \cos^2\alpha_2. \tag{6.17}$$

By solving Equation 6.11, an expression for the altitude as a function of position x and y is obtained as

$$z(x,y) = \frac{-B + \sqrt{B^2 - 4AC'}}{2A} \tag{6.18}$$

and

$$C' = C - \frac{\Delta\phi(x,y,\alpha_1,\alpha_2)}{2\pi}\lambda. \tag{6.19}$$

Sample 2D ISAR images as well as the combined 3D image of a complex aircraft are shown in Figures 6.4 [9] and 6.5 [8] for research done at the Beijing Institute of Environmental Features. The two figures are not images of the same aircraft, but instead were created on two separate occasions. For the second experiment shown in Figure 6.5, wideband signals are fed from one antenna and received by two antennas at different altitudes. Figure 6.5c shows that it is possible to correlate image scattering centers to physical geometry on the aircraft. Moreover, a slight roll can be perceived in the aircraft.

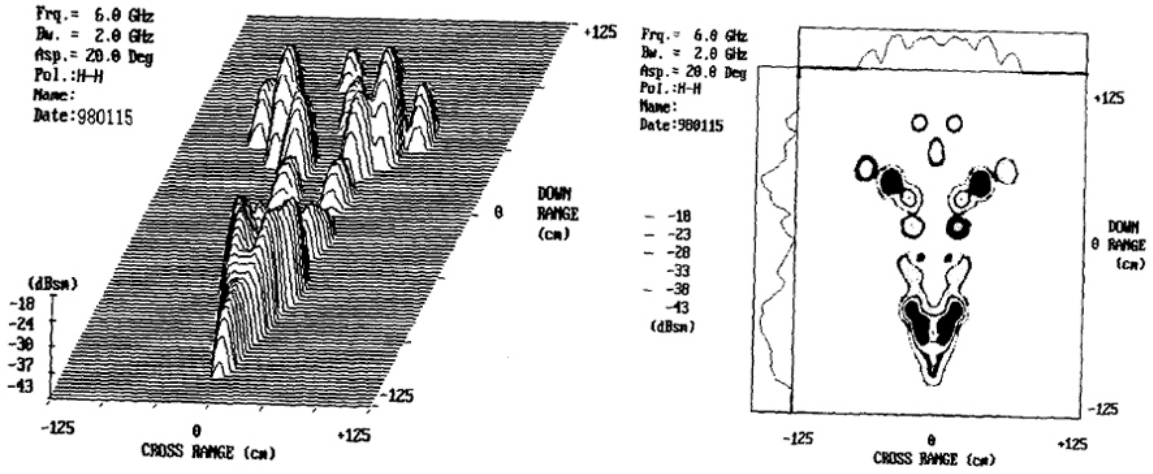


Figure 6.4 Interferometric 3F ISAR Images (From Ref. [9].)

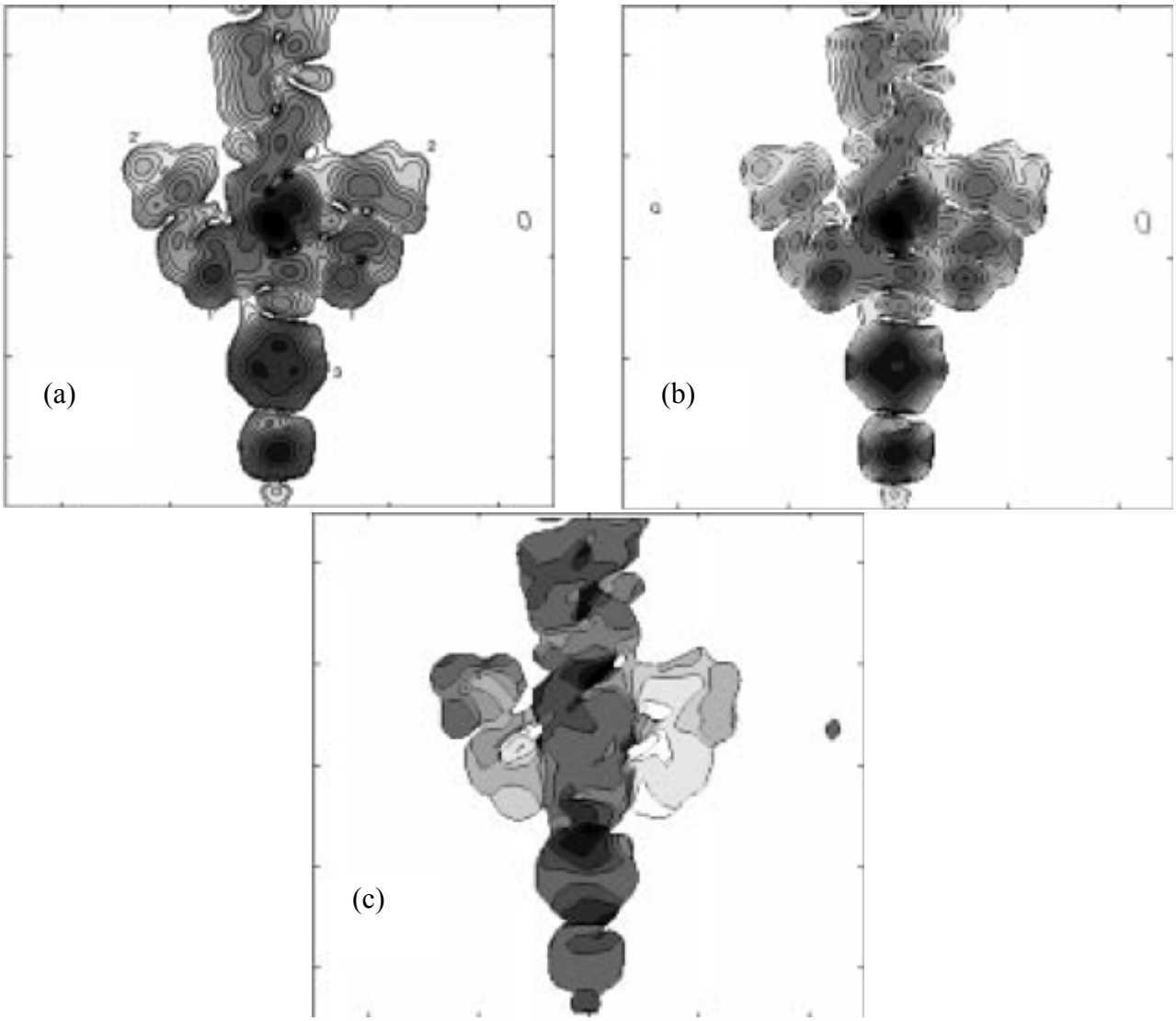


Figure 6.5 Interferometric 3D ISAR Images(From Ref. [8].)

In general, interferometric ISAR imaging provides a fairly reliable method of extracting the third dimension of altitude from 2D ISAR images. Two shortcomings of this approach include the phase unwrapping limitation previously mentioned as well as difficulty in resolving multiple scatterers within a single range cell [8].

C. 3D ISAR USING THREE RECEIVERS

The three-receiver approach to 3D ISAR is similar in concept to the two receiver interferometric approach. As Figure 6.6 shows, three antennas are aligned in the same plane, yet along orthogonal axes. The transmitting antenna is at the origin, which is also

a receiver, and one antenna is placed along both the y-axis and z-axis. The slant-range and cross-range dimensions for each antenna should be identical. Therefore the phase shift at the three orthogonal points relative to each other provides the extra information needed to extract an elevation from the traditional 2D ISAR signal returns. The goal is to use the phase information to correctly judge the cross-range scale of each image plane. Once the real scale factor is known, an accurate correlation between the image planes is made.

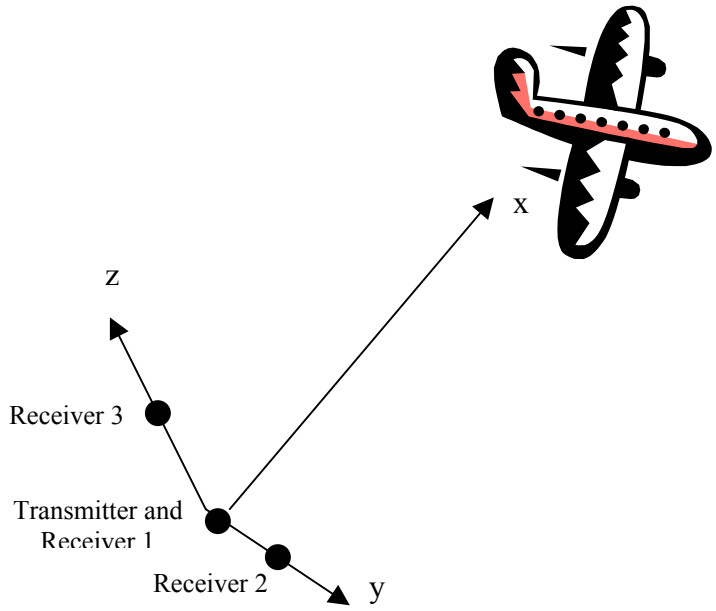


Figure 6.6 3 Receiver Radar Setup (From Ref. [10].)

Following the development in [10], this approach uses a Linear Frequency Modulated signal given by

$$s(\tau_r) = A_s e^{j2\pi(f_c\tau_r + \frac{k}{2}\tau_r^2)}, \tag{6.20}$$

from which a complex return signal is received as

$$s_{m,P}(t_r, \tau_r) = A_P e^{j2\pi\left[f_c(\tau_r - t_{md}) + \frac{k}{2}(\tau_r - t_{md})^2\right]} \tag{6.21}$$

where

A_s is the signal amplitude,

A_P is the amplitude of point scatterer P at the m-th receiver,

f_c is the carrier frequency,

τ_r is the fast time for the range dimension,

t_r is the slow time for the cross range dimension, and

t_{md} is the fast time delay at slow time instant t_r.

The distance from the m-th receiver to a point scatterer P at a time t_r can be related for each antenna as

$$R_{1P}(t_r) = \left\{[x_P + \Delta R(t_r) + \Delta R_{xP}(t_r)]^2 + [y_P + \Delta R_{yP}(t_r)]^2 + [z_P + \Delta R_{zP}(t_r)]^2\right\}^{1/2} \quad 6.22$$

$$R_{2P}(t_r) = \left\{[x_P + \Delta R(t_r) + \Delta R_{xP}(t_r)]^2 + [y_P + \Delta R_{yP}(t_r) + b]^2 + [z_P + \Delta R_{zP}(t_r)]^2\right\}^{1/2} \quad 6.23$$

$$R_{3P}(t_r) = \left\{[x_P + \Delta R(t_r) + \Delta R_{xP}(t_r)]^2 + [y_P + \Delta R_{yP}(t_r)]^2 + [z_P + \Delta R_{zP}(t_r) + b]^2\right\}^{1/2} \quad 6.24$$

where

x_P, y_P, z_P is the location of point scatterer P in each dimension,

$\Delta R(t)$ is the distance change from target to receiver due to translational motion,

b is the distance from receiver to transmitter, and

$\Delta R_{x,y,zP}(t)$ are the shifts of scatterer P from time instant 0 to time instant t_r in each dimension.

In order to properly align all three return signals, both a range-alignment and an auto-focus process must be used. If the range resolution of the radar system is less than 0.5 m, then the distances between receivers can be neglected and the same motion compensation functions used for all three antennas. After alignment and focusing, the received signal of all scatterers in the n-th range cell from the m-th receiver can be described by

$$S_{n,m}(t_r) = \sum_{P \in nth\_rangecell} A_P e^{-j\varphi_{m,P}} e^{-j\phi_P(t_r)}, \quad 6.25$$

and

$$\phi_P(t_r) = \frac{4\pi(x_P \Delta R_{xP}(t_r) + \Delta R(t_r)\Delta R_{xP}(t_r)}{R_P \lambda} \tag{6.26}$$

where $e^{-j\varphi_{m,P}}$ contains the position information of the scatterer that will be resolved in three-space and $R_P = \sqrt{x_P^2 + y_P^2 + z_P^2}$. Expanding the terms $\Delta R(t)$ and $\Delta R_{xP}(t)$ leads to the Doppler frequency of the P-th scatterer to each *m*-th receiver,

$$D_{m,P}(t_r) = \frac{d\phi_P(t_r)}{dt} = g_1(t_r) + g_2(t_r), \tag{6.27}$$

with

$$g_1(t_r) = \frac{4\pi\{x_P[(y_P - y_0)w_z(t_r) + (z_P - z_0)w_y(t_r)]\}}{R_P \lambda}, \tag{6.28}$$

and

$$g_2(t_r) = \frac{4\pi\{\Delta R(t_r)[(y_P - y_0)w_z(t_r) + (z_P - z_0)w_y(t_r)] + v(t_r)\Delta R_{xP}(t_r)\}}{R_P \lambda} \tag{6.29}$$

where

$v(t_r)$ is the translational velocity of the target,

$w_{y,z}(t_r)$ is the angular velocities of the target in the *y* and *z* directions, and

x_0, y_0, z_0 is the reference scatterer location chosen as focus point of target.

Further processing, including joint time-frequency analysis techniques, will eliminate the $g_2(t_r)$ term entirely [10] so that the resolved Doppler frequency of each scatterer for any given receiver is equal to Equation 6.28.

Sample 2D images of the three receiver ISAR approach are given from two different image planes in Figures 6.7 and 6.8. The algorithms just described present the general form of the processing sequence needed to create each properly motion corrected and resolved 2D image that are combined to form a 3D image. Depending on whether or not the target is maneuvering or somewhat stationary, application specific algorithms

have also been developed to increase the resultant image resolution and decrease the effects of multiple scatterers in a single range cell. All of the images shown and algorithms behind them come from research done at the University of Delaware, Newark [10].

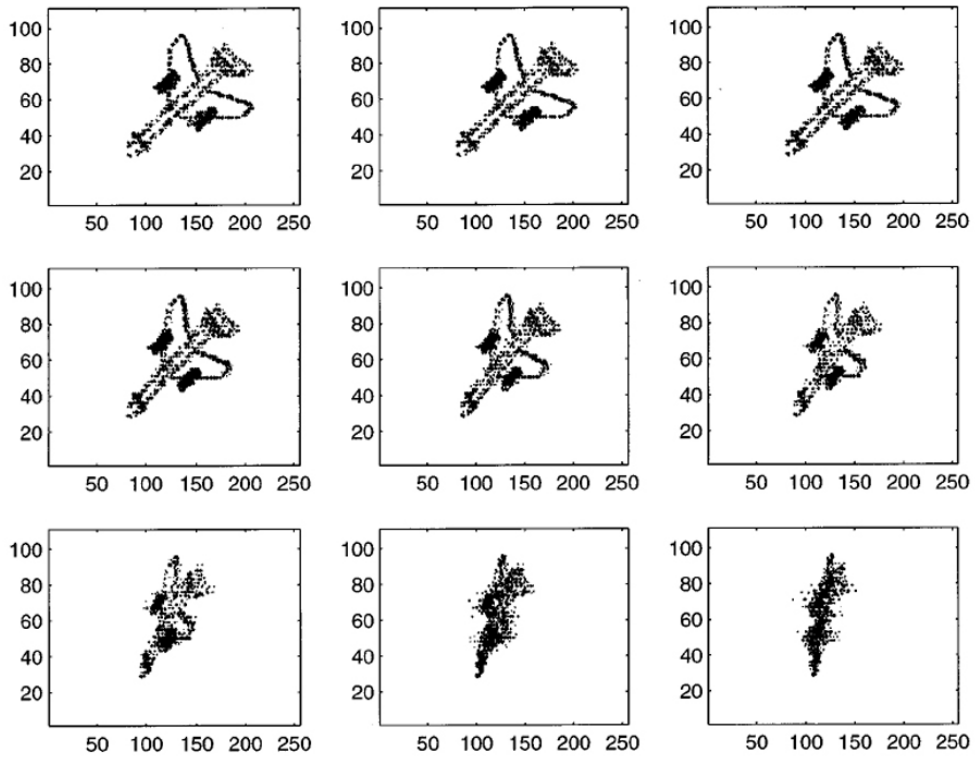


Figure 6.7 3 Receiver ISAR Image Plane (From Ref. [10].)

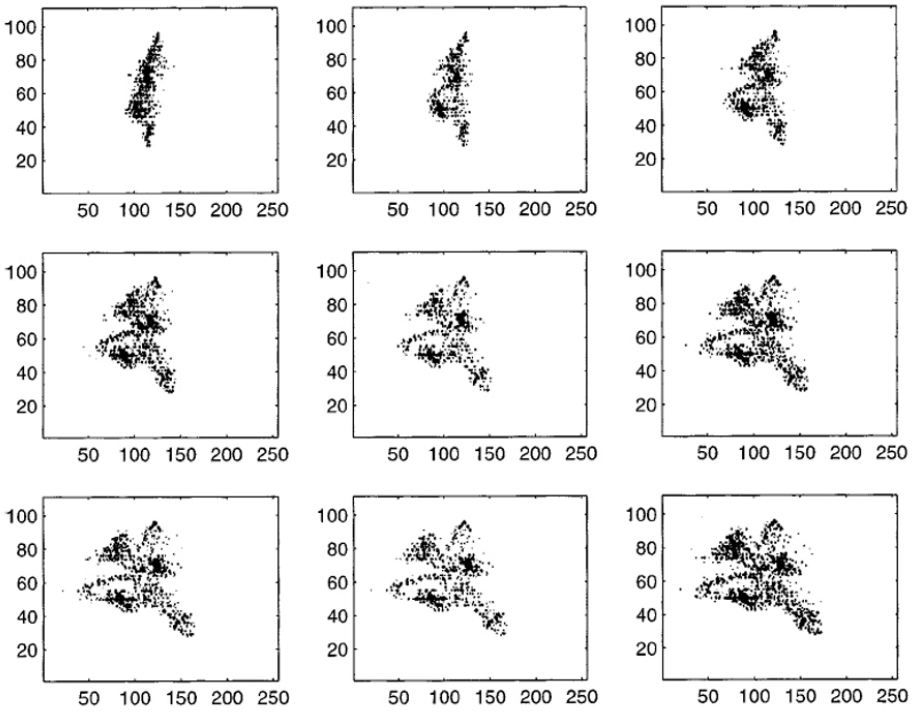


Figure 6.8 3 Receiver ISAR Image Plane (From Ref. [10].)

D. 3D DIS MODIFICATIONS AND CONCLUSIONS

3D ISAR is a fairly new concept in the world of high-resolution radar imaging. There are several approaches currently available, of which three have been briefly presented in this chapter. All of the approaches have their individual strengths and weaknesses. The motivation for creating a 3D ISAR image of a moving target is well founded in numerous military applications ranging from enhanced feature recognition on targets to simply providing more insight into current 2D ISAR imaging and feature correlation. One problem that is encountered for all approaches thus far is the ability to resolve multiple scatterers within individual range cells. Despite the sufficiently high radar resolution of current wideband systems, determining an elevation dimension from back-scattering effects in corresponding range cells becomes increasingly difficult at long range or for very large targets as seen in both the case of interferometric processing and the three-receiver image plane correlation approach. Despite the shortcomings of these techniques, they have shown that it is possible to construct 3D ISAR from current 2D images taken at varying altitude levels and with multiple receivers.

Since the full realization of a 3D ISAR is on the horizon at some point in the near future, it makes sense to take another look at the current 2D DIS system and suggest

ways that it might be modified to accommodate the generation of false 3D ISAR targets. It is important to again mention that two of the three 3D techniques discussed involve the use of multiple 2D images to form the 3D superposition of a target's reflections. Instead of rethinking the entire DIS process for a 3D ISAR countermeasure, one must only examine the current 2D images that are created and determine what, if any, backscattering effects are already included in the phase information of these signals.

It is therefore important to guide future research in finding the best way of combining multiple 2D ISAR images created by the DIS, based upon the techniques already described, to form a 3D ISAR superposition. Modifications to the current 3D techniques might include various processing changes to accommodate the chirp-pulse format of current ISAR systems such as the AN/APS-137 and AN/APS-147. In particular, hardware changes to the DIS may be limited to phase modulation changes. The extracted phase coefficients must be analyzed to ensure that they retain the necessary backscattering dimensions important to 3D reconstruction algorithms. More importantly, the phase coefficients must take into account the fact that multiple receivers are going to analyze the new false-target image signal and thus coefficient creation should not be based simply on the results obtained from one receiver. The next chapter presents current 3D graphics software tools that aid in this process by providing fully immersive environments where current 2D ISAR images are overlaid on top of the geometric model of the corresponding ship being imaged.

E. SUMMARY

This chapter introduced the concepts of 3D ISAR as found in current research efforts. In particular, 3D monopulse radar, 3D interferometric ISAR, and three-receiver ISAR were each presented along with their individual strengths and weaknesses. Based on this research, modifications to the current DIS were suggested for future processing and creation of 3D ISAR return signals. Chapter VII provides details on the visualization tools that were used to further explore the 3D ISAR concept.

VII. 3D GRAPHICS VISUALIZATION TOOLS

The algorithms developed and discussed in the previous chapters for both 2D and 3D ISAR provide a sound mathematical basis and motivation for continued work in the realm of 3D high-resolution radar. These processing techniques and their corresponding waveform data present a wealth of knowledge about current radar imaging techniques as well as insight into the next generation of radar platforms for use in military operations. However, much of this valuable information is encrypted in the massive data sets themselves. Without adequate methods of visualizing these results, the full impact of their discovery is lost. To be able calculate a 3D ISAR image, but not display it defeats the very purpose of its calculation. This chapter presents currently available tools that have the ability to fully immerse the user in a three-dimensional virtual realm wherein data of all sorts may be fully visualized and analyzed for completeness. The basis for this world is the Virtual Reality Modeling Language (VRML), harnessed for use within the open-source X3D-Edit software. An overview of this software is presented, followed by a discussion of 3D radar implementations within X3D-Edit. The second part of the chapter deals explicitly with new tools created as an integral part of this thesis to provide Non-Uniform Rational B-Spline (NURBS) support to the X3D and VRML environments. This fully functioning implementation within the X3D and Java domain adds the capability to develop smooth 3D curves and surfaces for use in both complex object creation and smooth animation paths such as those for flight trajectories. Through the combination of X3D and NURBS, ISAR data can be fully visualized to aid in future developments of 3D radar imaging.

A. X3D-EDIT AUTHORING TOOL

Extensible 3D (X3D) is an open standard file format and programming platform for developing 3D web-based graphics. Its roots lie in the VRML standard created in 1997 which specifies how shapes, materials, objects, sound, and animation all come together to present a fully interactive 3D world, viewable on most standard internet browsers. However, X3D is a fully developed specification approved by the International

Standards Organization (ISO) [11] that brings VRML concepts to an entirely new level. As its name suggests, it is built on Extensible Markup Language (XML) foundations and is thus fully extensible in nature, allowing it to reach into new avenues of web-based productivity. X3D has thus far been extended to support everything from TCP/IP sockets across a network to communicating with MATLAB functions. Aside from being a close relative of XML, X3D can also be extended through Java calls to provide the full capability of a true programming language to the user, thus opening up a realm of practically limitless extension possibilities. X3D-Edit, developed in part at NPS, provides an intuitive yet powerful means of creating virtual environments within the X3D specification. Figure 7.1 illustrates the graphical user interface (GUI) for X3D-Edit. This interface is based upon the Xeena platform, developed by IBM [12].

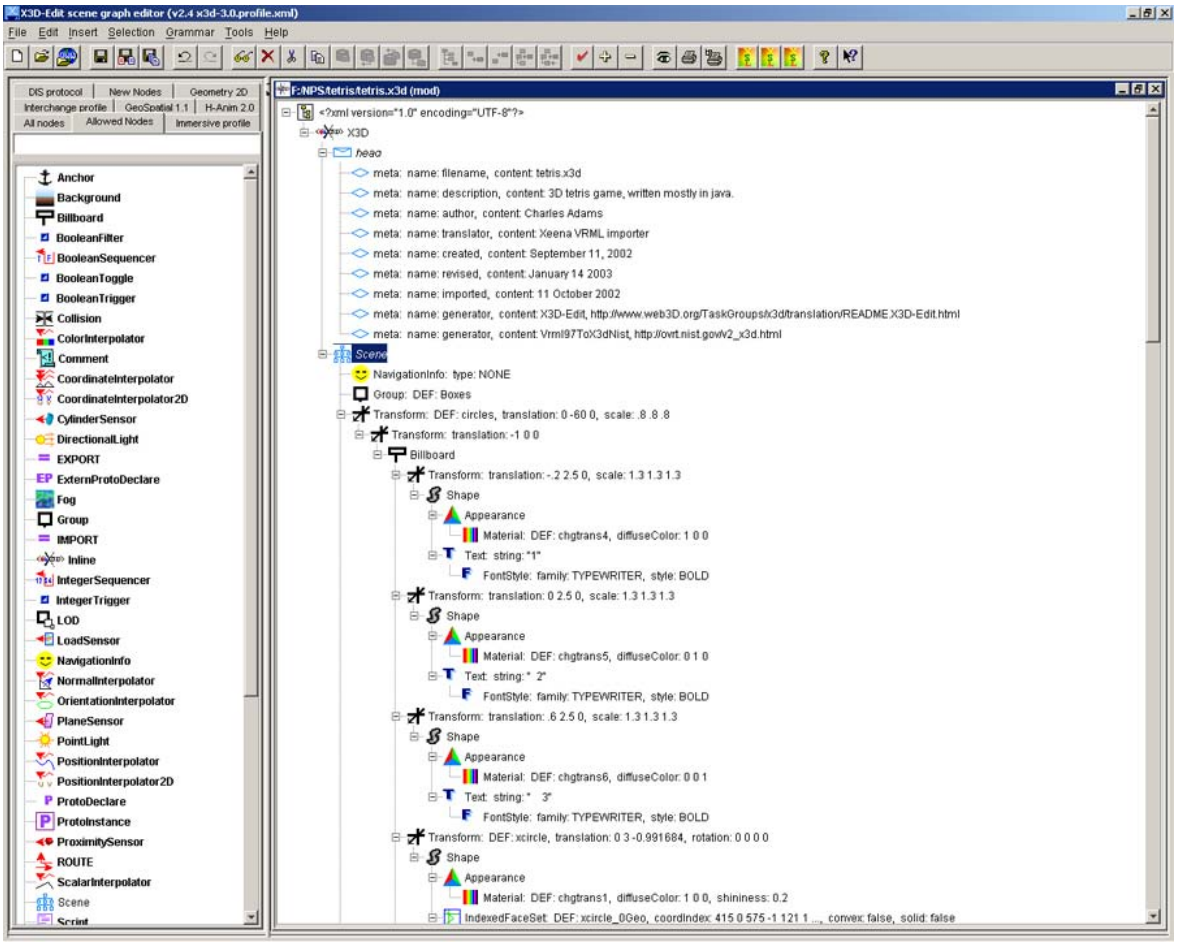


Figure 7.1 X3D-Edit GUI.

The underlying theme behind X3D-Edit is modular interconnectivity. Each aspect of a virtual world from simple primitive shapes to lighting and sound is considered a node. Each node will have various parameters describing how it behaves within the virtual world. Moreover, nearly all nodes have at least one input field and output field through which they may communicate with either the browser or another node. A simple scene might consist of a Shape node whose child is an IndexedFaceSet geometry node that provides the browser with a complete description of a 3D model such as an airplane. Also attached to the Shape node may be an Appearance node that also includes a Material node to describe the airplane's surface color and texture. Among the more complex nodes included in X3D, and thus also in X3D-Edit, are PositionInterpolator nodes, TimeSensor nodes, and Script nodes. Through combining these nodes it is possible to quickly create interactive 3D content. The Script node is especially important as it allows the user to add either Javascript or full Java programming to the scene. Figure 7.2 shows a screen shot from a scene authored almost entirely in Java. Since the Java is connected to X3D via a Script node, it is afforded the luxury of using X3D's inherent geometry and graphical navigation support. X3D simply serves as a display tool, or graphics engine, for high-level code to manipulate. In the case of Figure 7.2, the high-level Java code creates a fully functioning 3D Tetris game.

Figure 7.2 Java 3D Tetris Game Within X3D-Edit.
(From http://www.web3d.org/TaskGroups/x3d/translation/examples/games/3dtetris.wrl.)

B. 3D RADAR VISUALIZATION

Using the tools of X3D-Edit, it is thus possible to help visualize what a 3D set of ISAR images looks like if created via the superposition of multiple 2D ISAR images. At the time of this thesis, the desired set of sample 2D ISAR data has not arrived from the Naval Research Laboratory. Therefore, this section gives a description of the desired implementation within X3D and shows a scenario based upon sample ISAR images. These images are formed from the same sample data as was used to evaluate the DIS Adder Overflow in Chapter IV.

The aim of this simulation was to recreate both the motion of the target ship and the motion of the radar platform aboard a P-3 aircraft. On top of the visible ship model is placed the 2D ISAR image corresponding to both the motion of the P-3 and of the ship.

In the fully immersive virtual world created for the simulation, the user is able to navigate the viewpoint camera as needed to examine the resulting image from all angles. The starting surface ship model used in the simulation is pulled from the online SAVAGE [13] model library of military vessels and is the model for a destroyer. The model for the P-3 was created for this simulation using primarily NURBS geometry. The smooth flight path of the P-3 is a smooth NURBS animation path. All NURBS geometry and animation paths in the simulation are implemented at runtime via the NURBS Java engines created in conjunction with this thesis and explained in more detail in subsequent sections.

The experiment is conducted in several stages. First, the basic layout of the scenario is created, including only the ship model, P-3, and single ISAR image. Aligning the image to the profile of the target ensures that the correct image aspect ratio and perspective are achieved. The setup is shown in Figures 7.3 and 7.4.

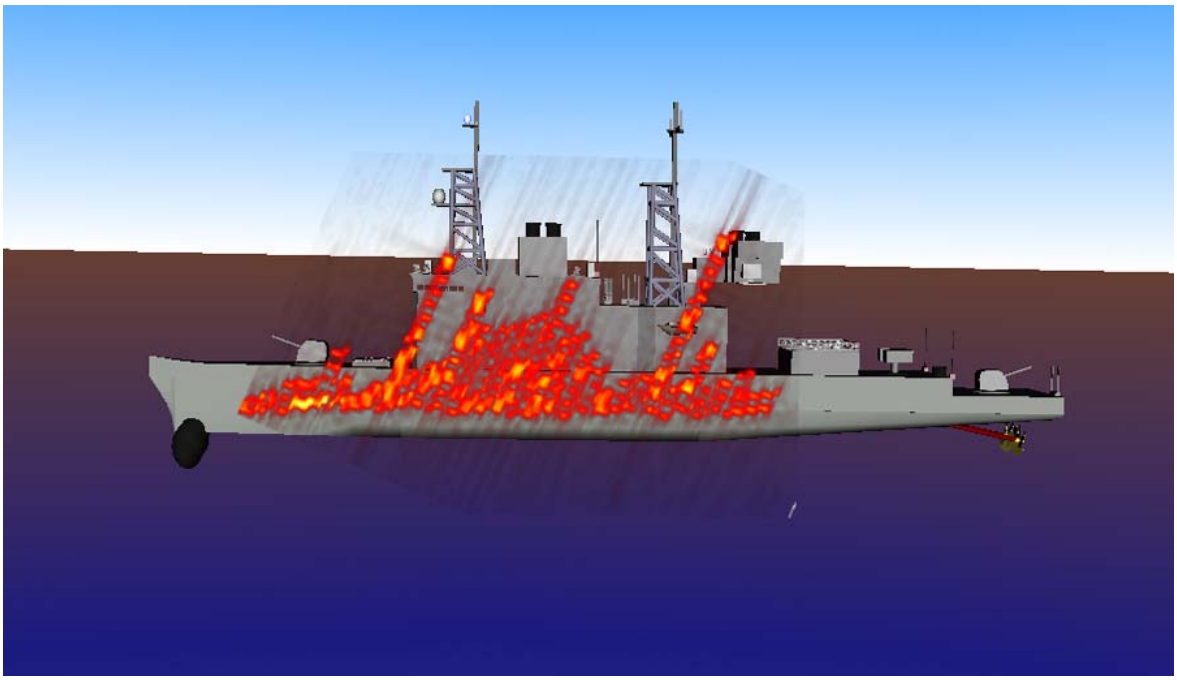


Figure 7.3 2D ISAR Overlay in X3D.

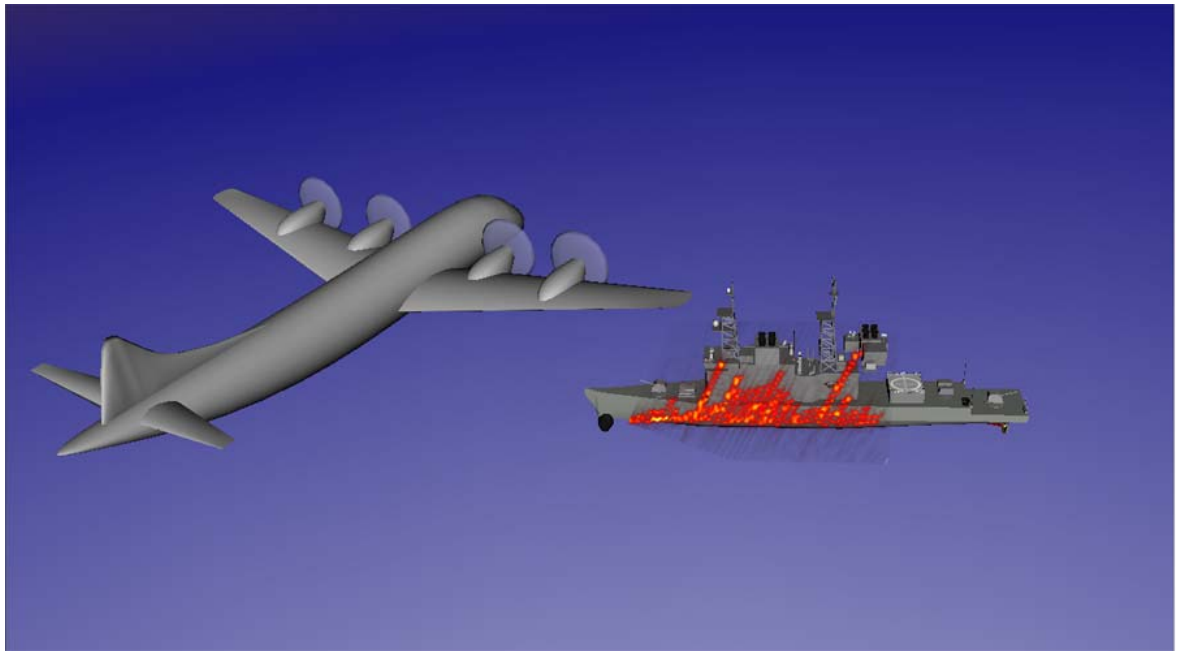


Figure 7.4 2D ISAR Overlay in X3D Stage 1.

By analyzing this simple scenario, it is evident that new information about 3D ISAR methods can be obtained if the simulation is extended to include a progression of images. The second stage in the experiment adds motion to the scene by correlating the movement and rotation of the target with the projected flight path of the P-3. Moreover, a sequence of 600 2D ISAR images, corresponding to twenty seconds of data, is created and overlaid upon the ship model. Figure 7.5 gives the corresponding setup. Notice that two different ship models are used from Figure 7.4 to Figure 7.5. In reality, the simulation data is created to resemble the profile of the USS Crocket, yet does not correspond exactly to any specific ship including the Crocket. The point is made that the simulation is flexible and thus able to use any target model through the simple interchange of geometry nodes. Once real ISAR data is received, choosing the specific vessel that was originally imaged becomes important. The goal of this experiment is to provide proof of concept.

.

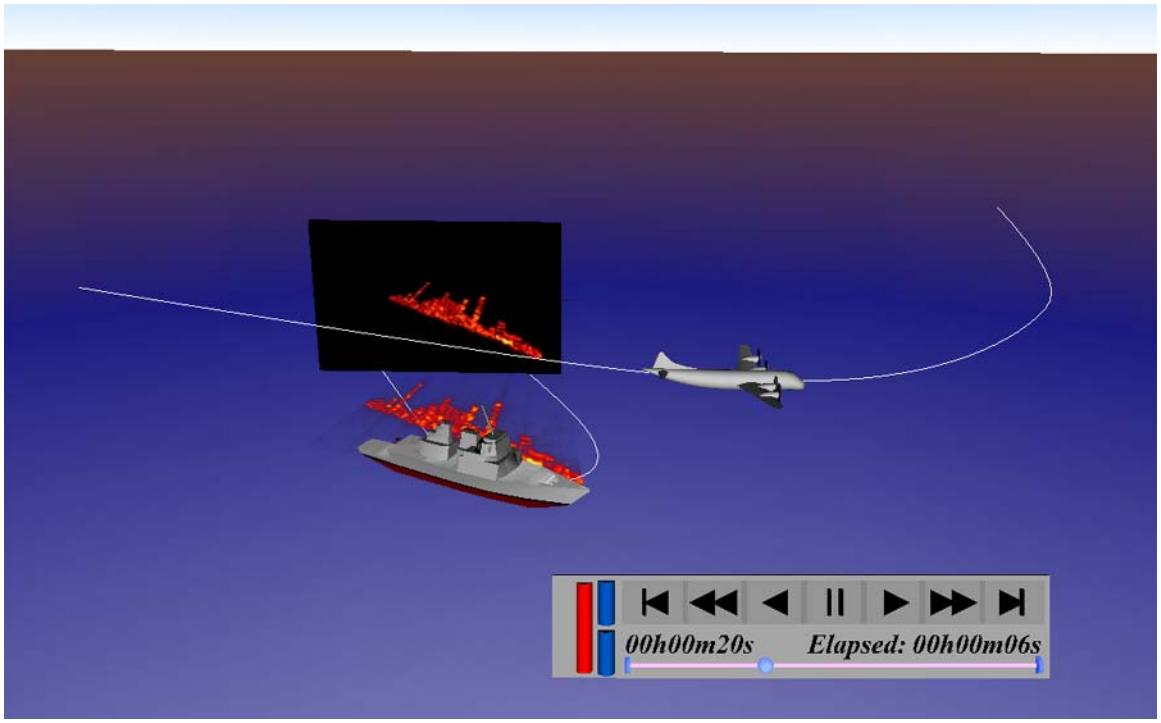


Figure 7.5 2D ISAR Overlay in X3D Stage 2.

This sequence is normally composed of real ISAR data recorded from an actual AN/APS-137. For simulation purposes, however, the sequence used is created manually through the modification of current range-Doppler profiles and in such a way as to emulate real ISAR data. Particularly, range-Doppler images are dependent upon both the angle of the radar to the target and the motion of the target. Therefore, the highest resolution image that can be obtained is achieved when the radar is 45 degrees off the bow of the target. When the radar deviates from this angle, the pitch of the ship is not as prominent with respect to the P-3 and thus lower Doppler shifts are received. Notice the difference in the magnitude of the range-Doppler images of Figures 7.6 and 7.7. In particular, Figure 7.7 represents an image of the target at a completely perpendicular side angle where there is little rocking back and forth of the ship. Within the simulation, a play-control menu allows the user to navigate between the beginning and end of the animation. Two image planes are overlaid on the target. One is located on the target itself and is transparent around the range-Doppler image. This image plane is controlled directly by the play controls in order to examine each frame of the animation. The other image plane is located above the target and provides a coherent movie of the image sequence for improved continuity across the length of the simulation.

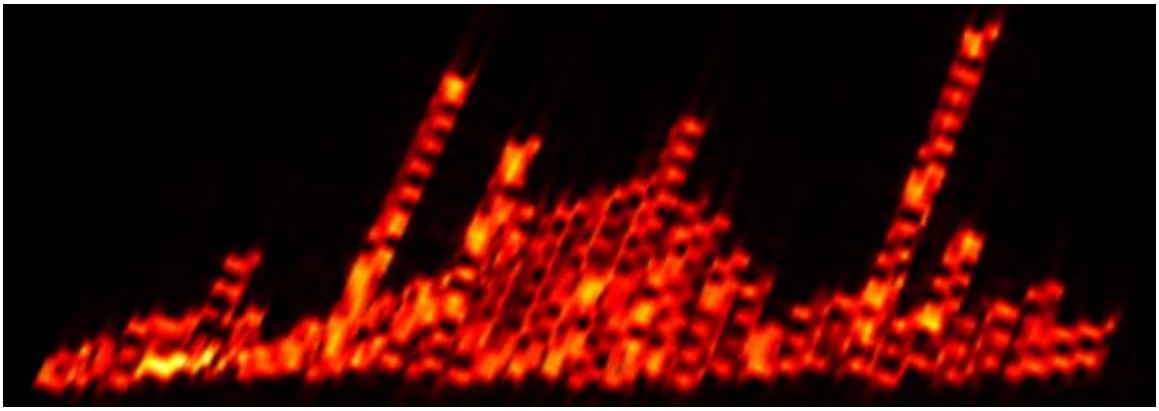


Figure 7.6 2D ISAR Range-Doppler Image in X3D with Large Doppler Shift.

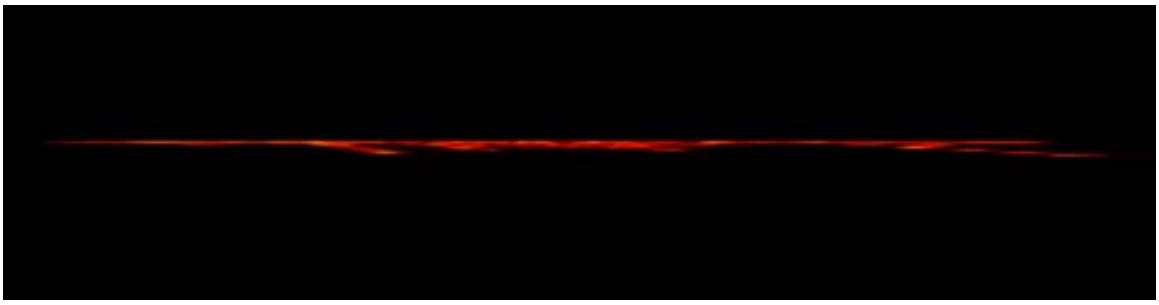


Figure 7.7 2D ISAR Range-Doppler Image in X3D with Small Doppler Shift.

From the previous figures, it is possible to align the geometry of the target with the Doppler shift within the image sequence. As the target rotates with respect to an effectively fixed radar platform, it traverses a full 90-degree angle from side to side. The range-Doppler image at each increment of this angle will exhibit a magnitude proportional to the angular velocity of the target. By collecting the images at each angular increment, a full 90-degree profile of the ship is obtained. Although some of the resulting images will be inverted due to the target rotating away from the platform, their magnitude may still be used to construct the original ship profile. This is the goal of stage 3. The approach taken is to begin with the set of 14 range-Doppler images, each created roughly 6 degrees apart corresponding to a target rotation from 0 to 90 degrees. Figure 7.8 shows four of the images in the set.

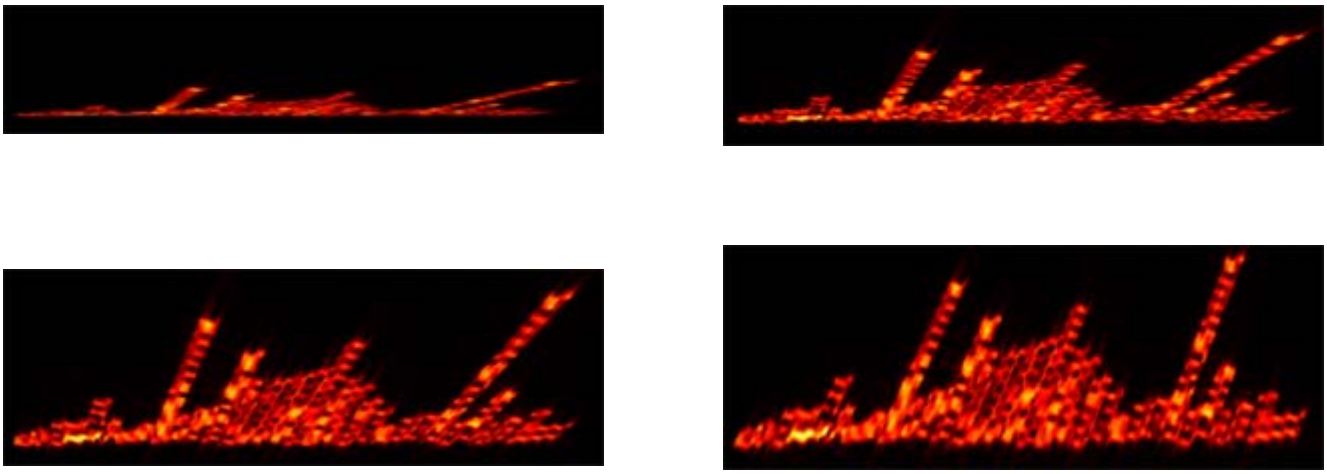


Figure 7.8 4 Sample 2D ISAR Range-Doppler Images.

These 14 images are subsequently overlaid on top of each other and separated uniformly in range. A transparency channel allows images in the back to show through the empty parts of foreground images. Figures 7.9 and 7.10 represent the superposition of images from two different viewing angles. By combining the images in X3D, it is possible to quickly maneuver the camera and view the set from all angles.

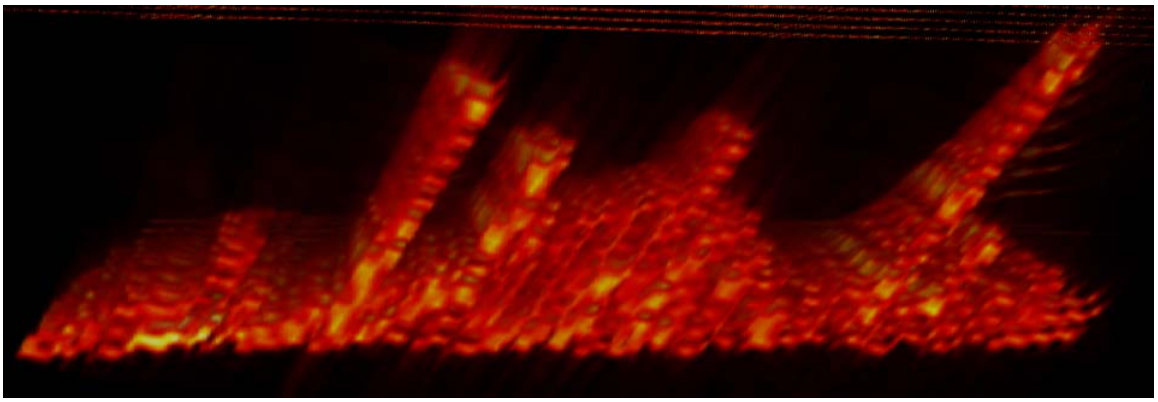


Figure 7.9 3D Range-Doppler Superposition.

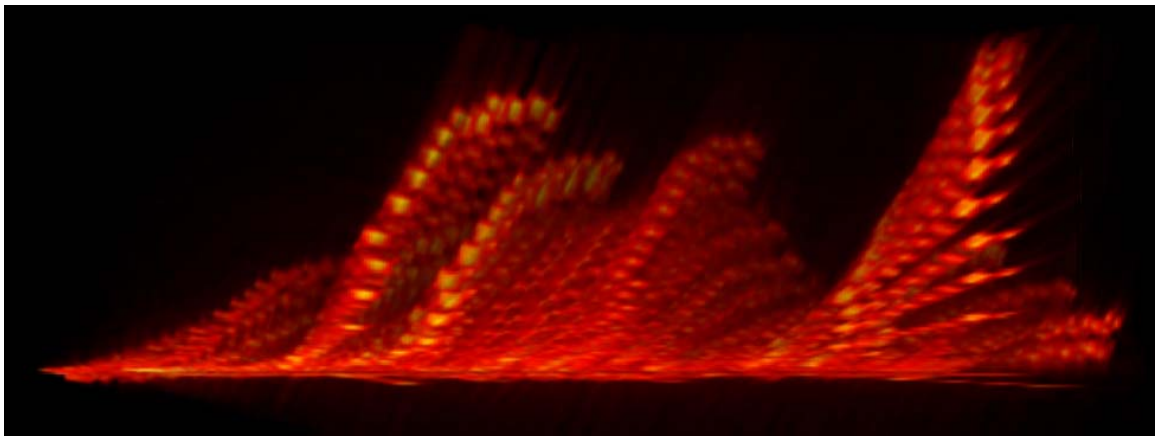


Figure 7.10 3D Range-Doppler Superposition.

The figures represent an actual example of how 2D images may be combined to form a 3D perspective. Although these sample images come from the DIS software simulation, this same approach can be used to combine 2D images obtained through interferometric ISAR or the 3-receiver ISAR setup. Being able to combine images from truly orthogonal receivers allows for the cross-range scale dimension, estimated in Figure 7.10, to be computed. Once the scale factor is determined, actual geometry can be formed from the multiple range profiles.

The generation of geometry is simulated in this experiment based upon the superposition of 2D sample images just discussed. Since each image represents a slice of the target profile at a specific angle, the creation of a geometric slice corresponding to each range profile allows for the superposition of these geometric slices to estimate target depth. This is done by creating profile curves for each of the 14 images. From the set of profile curves, a set of 14 extrusion surfaces is generated. Coherently aligning each extruded surface and offsetting each profile in the range dimension, a 3D representation of the original target is obtained. Figure 7.11 depicts a linear approximation between geometric profiles while Figure 7.12 shows a cubic approximation, representing the overall surface curvature of the return signals.

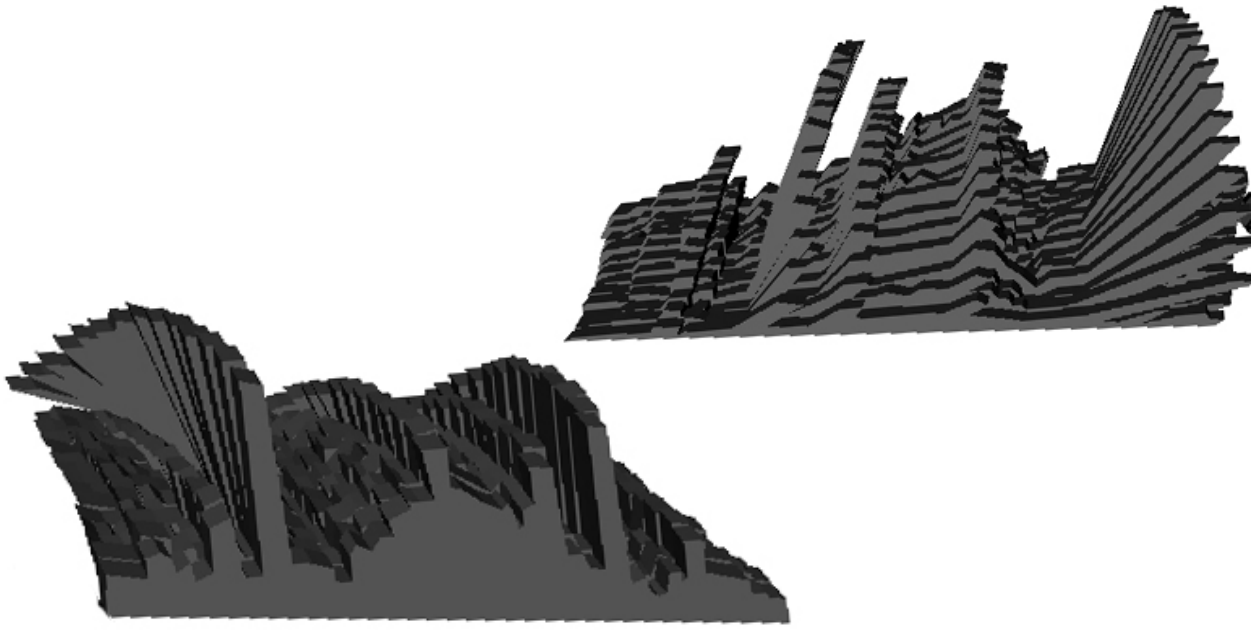


Figure 7.11 3D Geometric Profile Linear Approximation.

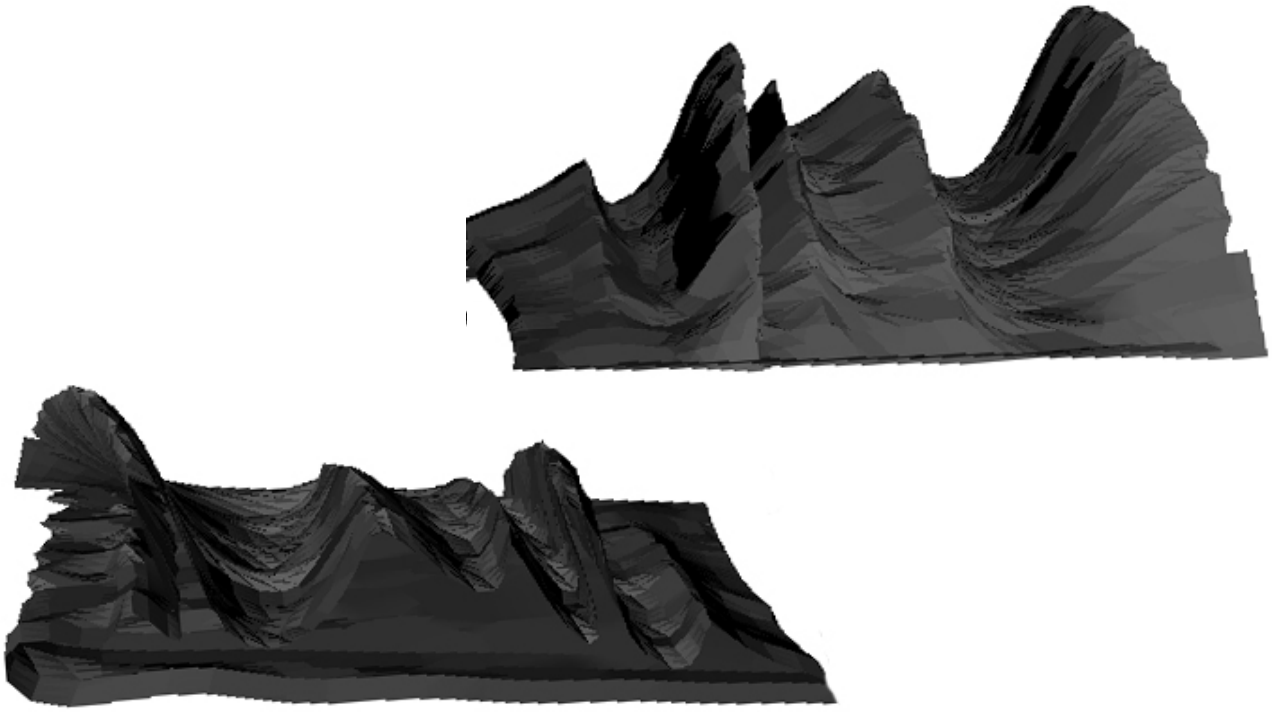


Figure 7.12 3D Geometric Profile Cubic Approximation.

The new profile geometry is further adjusted to fit the profile of the original target. Moreover, since the target is symmetric, the geometry can be copied and reversed to create a complete 3D ISAR geometric model. Figure 7.13 shows this final model.

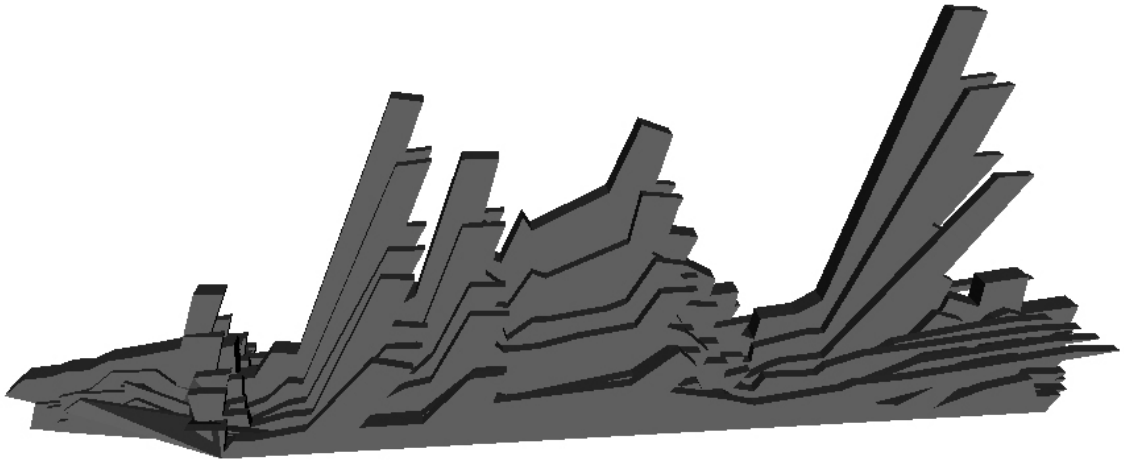


Figure 7.13 3D Geometric Profile Model.

Once the new 3D geometric model is obtained from 2D ISAR images, the original simulation is run once more. This time, the range-Doppler profiles are overlaid on the new geometric model to validate the creation process. Figures 7.14 and 7.15 show screenshots from this validation.

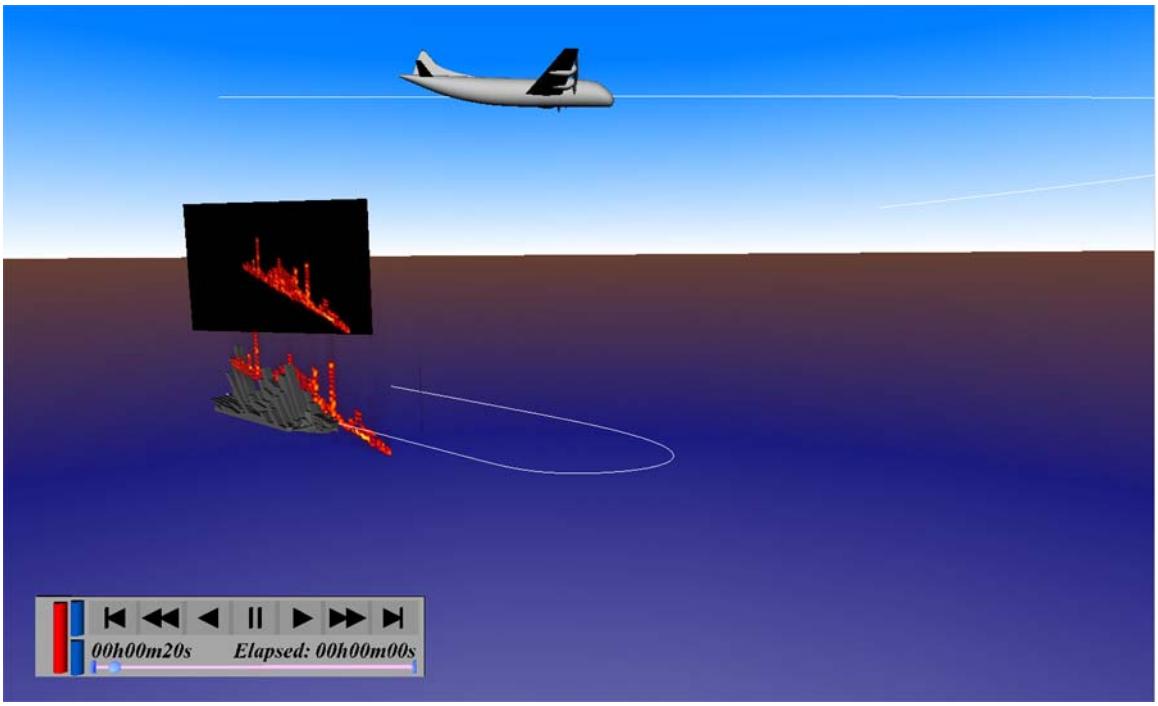


Figure 7.14 Final 3D ISAR Overlay Simulation.

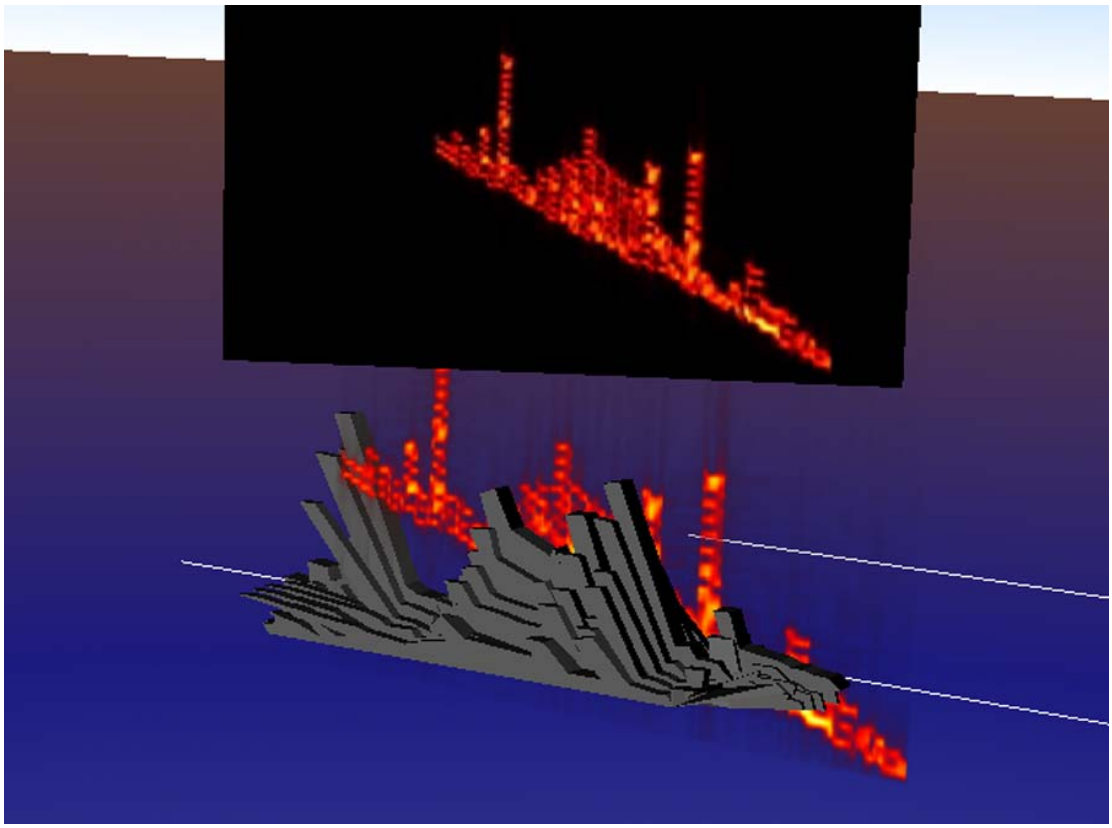


Figure 7.15 Final 3D ISAR Overlay Simulation.

Through the combination of a fully immersive 3D environment and geometric estimations of range-Doppler profiles, the user is able to visualize all angles of the 3D ISAR generation process. This approach sheds new light on developing 3D ISAR techniques and forces researchers to explore new avenues of study, including similar geometric simulations using data received from the techniques suggested in Chapter VI.

C. NON-UNIFORM RATIONAL B-SPLINES (NURBS) OVERVIEW

Computer graphics is rooted in the ability to mathematically describe reality. Graphics hardware knows nothing of what a sphere looks like and cannot comprehend a line segment with an infinite amount of precision. Like all concepts within a digital domain, shapes are simply estimations of reality based upon a mathematical description and implemented in a finite memory space. In the world of computer graphics, polygons are typically the fundamental building blocks of reality itself. Simple triangles combine to form both the smallest detail and smoothest curve of a company's latest sports car within Autocad.

Nevertheless, to describe the position of each polygon along that smooth curve, one must be able to describe the path, or curve itself, along which the shape follows. The simplest way to describe a curve is to create a piece-wise linear set of line segments such as Figure 7.16

.

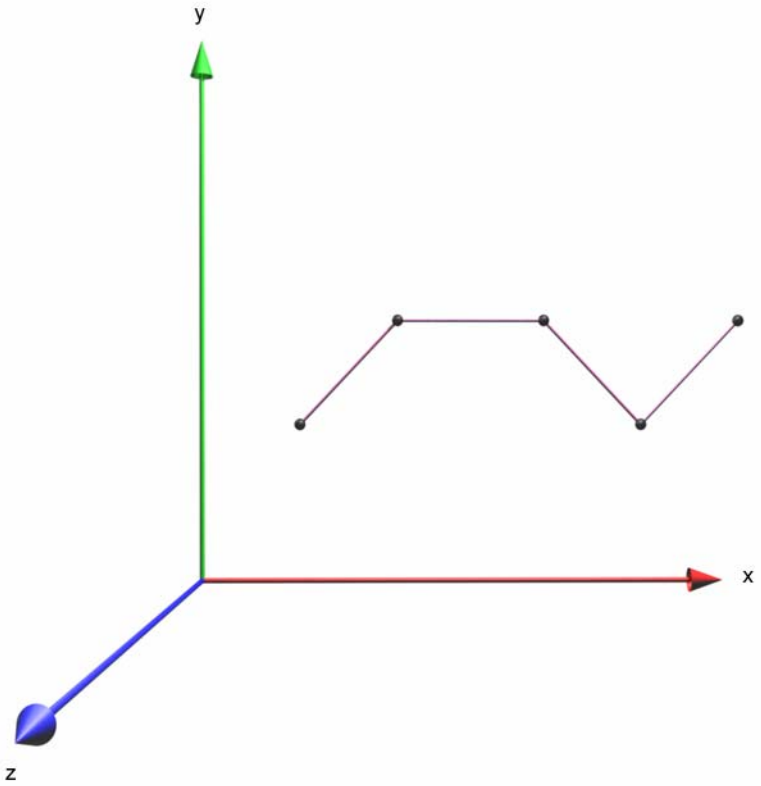


Figure 7.16 Piecewise Linear Curve.

The shortcomings of this approach are rather obvious, producing a path with hard corners and a generally "blocky" appearance. Over the past fifty years, many researchers have developed sets of equations used to both describe and manipulate a set of points through which a smooth curve may be drawn. This chapter summarizes the basis for the representation of curves and surfaces using Non-Uniform Rational B-Splines (NURBS). Although not a new concept, NURBS have been distinctly lacking in the implementation of X3D software. In order to create the necessary geometry, including a NURBS P-3 aircraft and its flight path, for use in 3D ISAR explorations corresponding to the current DIS system, special effort is taken to implement the NURBS specification for X3D. This section details the process of that undertaking.

1. The Bézier Curve

A curve is simply a continuous path from one point to another. It may be open or closed, straight or twisted. In each case, there is a specific starting point and end point, between which a path is created. The object of representing this curve within a limited memory space, such as a graphics card's hardware buffer, is to be able to describe the curve without loss of precision by as few points as possible. French engineer Pierre

Bézier, working for Renault Automobile, developed some of the initial theories behind free-form curves and surfaces.

Bézier's general approach is to use a control polygon whose vertexes, or corners, govern the shape of the curve between them. Following the development in [14], the mathematical model that describes how the curve behaves between these corners is known as the curve's basis. For Bézier curves, the basis used is the Bernstein Basis, whose form is shown below.

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \qquad 0 \le t \le 1 \tag{7.1a}$$

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad (0)^0 \equiv 1 \tag{7.1b}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \qquad 0! \equiv 1 \tag{7.1c}$$

where

$P(t)$ is the point along the curve corresponding to the parametric parameter t,

n is the degree of the curve,

B_i is the set of control points defining the control polygon, and

$J_{n,i}(t)$ is the Bernstein Basis for the i-th control point corresponding to a degree of n.

Several important properties of Bézier curves include:

- The basis functions are real
- The degree of the polynomial defining the curve segment is one less than the number of control polygon points, or order of the curve
- The curve generally follows the shape of the control polygon
- The curve travels through the first and last control points

- The order of the curve is equal to the number of control points on the control polygon.

The Bézier version of Figure 7.16 is shown in Figure 7.17. Five points define the degree-four curve. Bézier curves are generally simple to understand. A higher degree, smoother curve is created simply by adding more control points to the control polygon. This, however, limits the effect that each control point can have on small, local regions of the curve. A change anywhere is felt throughout the entire curve. Moreover, a long curve, composed of many points can become computationally expensive to generate when the degree of the curve is inseparable from the number of control points.

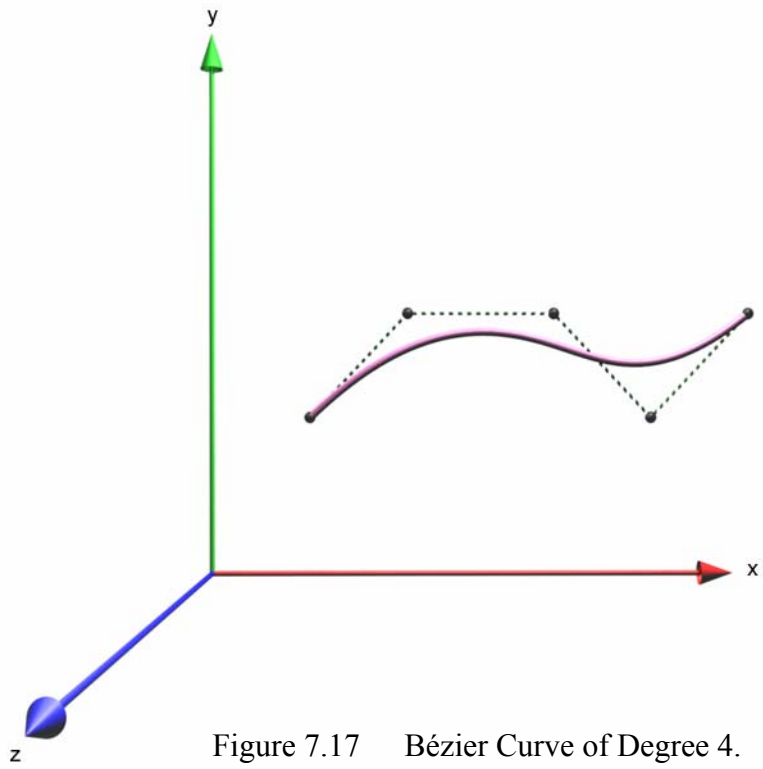


Figure 7.17 Bézier Curve of Degree 4.

2. B-Splines

In order to gain more freedom within the curve, a more localized basis function can be used. B-Spline curves begin with the Bézier approach of using a control polygon to define the bounds of the curve. Similarly, the parametric parameter t describes the position vector along the curve. The format of a B-Spline is given as [14]

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t) \qquad t_{\min} \le t < t_{\max} \qquad 2 \le k \le n+1, \tag{7.2a}$$

with

$$N_{i,1}(t) = \begin{cases} 1 & \text{if} \quad x_i \le t < x_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{7.2b}$$

and

$$N_{i,k}(t) = \frac{(t - x_i) N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t) N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{7.2c}$$

where

$N_{i,k}(t)$ is the Cox-de Boor Basis for the i-th control point,

x_i is the i-th element of a knot vector, and

$k = n+1$, is the order of the curve.

There are still $n+1$ control points, represented as B_i. The difference is that $N_{i,k}(t)$ is defined by the recursive Cox-de Boor formula where x_i represents elements of a knot vector. Properties of B-Splines are generally the same as for Bézier curves. Some of the more important properties include [14]

- The sum of the basis functions for any value t is 1,

$$\sum_{i=1}^{n+1} N_{i,k}(t) \equiv 1 \tag{7.3}$$

- Each basis function is positive or zero for all parameter values
- Transformations are applied to individual control polygons as opposed to the entire curve.

The most important part of a B-Spline is the knot vector, x. The knot vector is simply a set of floating point values, equal in number to the order of the curve plus the number of control points. It is composed of non-decreasing values that span the length of the curve. In other words, it is possible to animate an object along a B-Spline by

animating the parameter t from the first knot value, linearly, to the last knot value. Moreover, the order of the curve is not bound to the number of control points in the curve except that the number of control points must be at least equal to the order. The last, and perhaps most powerful property of a B-Spline, is that changes to a control point are localized within the curve. This is especially useful in animation where computing the entire curve at each frame is not practical. Visually, B-Splines look nearly identical to Bézier curves. As such, the concepts of uniform and rational curves will be developed first before presenting examples of B-Splines.

a. Uniform vs Non-Uniform B-Splines

Thus far, B-Splines have been introduced as a better alternative to Bézier curves in complex designs due to the locality of each control point. Yet, under the B-Spline umbrella, several varieties exist. Figure 7.18 shows two examples of knot vectors. Since the knot vector determines how a curve will react to changes made to the control points, a uniform knot vector allows for a uniform distribution between control points. Although a detailed discussion of knot vectors and their impact is beyond the scope of this section, it suffices to mention that knot vectors are not intuitive. The knot vector shown in Figure 7.18a is considered uniform because its values are evenly spaced. By changing the spacing between these values, a non-uniform knot vector results as in the case of Figure 7.18b. Notice the change in curve concavity made by the non-uniform knot vector.

a. [0 1 2 3 4 5 6 7 8 9]

b. [0 0 0 5 9 10 30 31 51 52]

Figure 7.18 (a) Uniform and (b) Non-Uniform Knot Vectors.

Having the ability to create a non-uniform knot vector allows the designer the freedom to unevenly weight portions of the curve. This may be done to form a closed, non-overlapping curve while maintaining the proper curvature. However, very few objects are actually created with a non-uniform knot vector.

Knot vectors are allowed to be "open" as well. Contrary to a simple, uniform knot vector, open knot vectors include a multiplicity of knot values at the beginning and end of the vector equal to the order of the curve. Figure 7.19a shows an example for a curve of order 3, degree 2. Notice how there are three zeros, followed by four uniformly spaced values, and then three fives. The specific range of theses values is not important, and thus knot vectors may be normalized as in the case of Figure 7.19b. In either case, the result on the curve is the same.

a. [0 0 0 1 2 3 4 5 5 5] **b. [0 0 0 .2 .4 .6 .8 1 1 1]**

Figure 7.19 (a) Open Uniform and (b) Open Uniform Normalized Knot Vectors.

3. Non-Uniform Rational B-Splines (NURBS)

The formula for a Non-Uniform Rational B-Spline is [14]

$$P(t) = \sum_{i=1}^{n+1} B_i^h N_{i,k}(t), \qquad 7.4$$

where B_i^h are the four-dimensional homogeneous control vertexes. Projecting the four-dimensional control points back into the three-dimensional space by dividing by the fourth homogeneous coordinate yields

$$P(t) = \frac{\sum_{i=1}^{n+1} B_i h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)}, \qquad 7.5$$

where h_i is the weighting applied to i-th control point. Notice how Equation 7.4 is almost identical to the formula representing B-Splines. The difference lies in B_i^h which represent four-dimensional homogenous vertices that are projected back into the three-dimensional realm. The fourth dimension is what causes the curve to be rational where h_i is simply an extra weighting value applied to each control point. By increasing the weighting on the control point, the closer the curve will be to passing through that control point. Typically, the weightings are ignored by setting all h_i's to a value of 1. $N_{i,k}(t)$ represents the same Cox-de Boor basis function as for B-Splines.

In summary, a NURBS curve represents a smooth, easily manipulated path capable of practically any degree, uniformity, and weighting, to describe nearly any shape known to man. Figure 7.20 gives an example of a NURBS curve of degree 3. The next section shows how to extend the general NURBS curve formula to create NURBS surfaces.

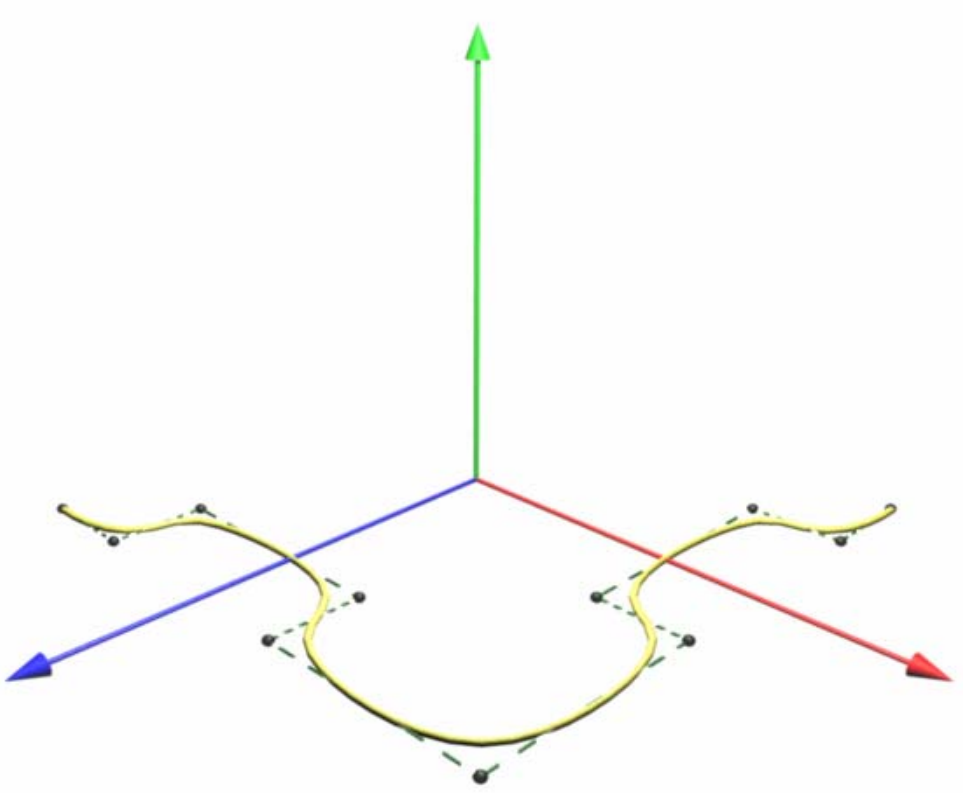


Figure 7.20 NURBS Curve of Degree 3.

a. Nurbs Surface

The logical extension of a NURBS curve is a NURBS surface. Simply put, a NURBS surface is nothing more than a patch created by a two-dimensional set of NURBS curves, represented by [14]

$$Q(u,v) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}^{h} N_{i,k}(u) M_{j,l}(v), \qquad 7.6$$

where

$Q(u,v)$ is the point on the surface corresponding to the parameters u and v,

$B_{i,j}^{h}$ is the four dimensional homogeneous control vertexes,

$N_{i,k}(u)$ is the Cox-de Boor Basis for the i-th control point corresponding to a degree of n in the u direction,

$M_{j,l}(v)$ is the Cox-de Boor Basis for the j-th control point corresponding to a degree of m in the v direction,

$k = n+1$, is the order of the surface in the u direction, and

$l = m+1$, is the order of the surface in the v direction.

The difference between the NURBS curve equation and this one is that the Cox-de Boor Basis functions are now calculated recursively for a double summation of orthogonal control points. If $N_{i,k}(u)$ represents the basis functions for the control points along one dimension, such as u, then $M_{j,l}(v)$ represents the basis functions for the control points along the v dimension, perpendicular to the u dimension. Consider Figure 7.21 for a visual example.

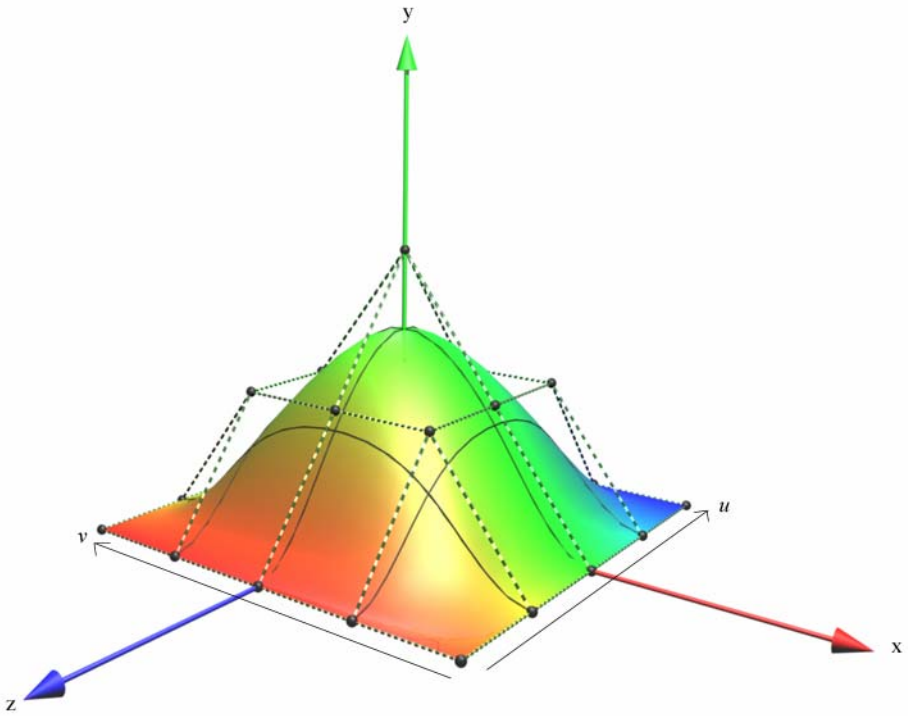


Figure 7.21 NURBS Surface.

Since there are two dimensions, there are also two knot vectors based on two different orders, one for the u direction and one for the v direction. It is not uncommon for a NURBS surface to have two different degrees. Moreover, nothing prevents one knot vector from being uniform while the other one is non-uniform. It is this flexibility in the mathematical model that gives NURBS surfaces such flexibility in practice.

D. X3D NURBS IMPLEMENTATION

The goal of implementing NURBS for X3D is to follow the X3D NURBS Specification created in 1997. Although the specification has been around for six years, the work done in this thesis represents the first open implementation of it for general-purpose use and distribution. Blaxxun Interactive and Parallel Graphics have both created proprietary implementations in the past. Yet these implementations are neither current nor consistent with the X3D specification.

The basis for the current open implementation lies in the ability of the X3D/VRML language to provide a Java interface to the X3D world. As such, there is an

X3D Prototype node created for each part of the NURBS specification that is implemented. Each Prototype node references a Script node that contains a full-fledged Java implementation of the math just discussed behind the NURBS node. This section describes the Java implementation for NurbsCurve, NurbsSurface, and NurbsPositionInterpolator Prototypes. The code base may be found in the Appendix.

1. Nurbs Curve Prototype

The NurbsCurve specification is shown in Figure 7.22.

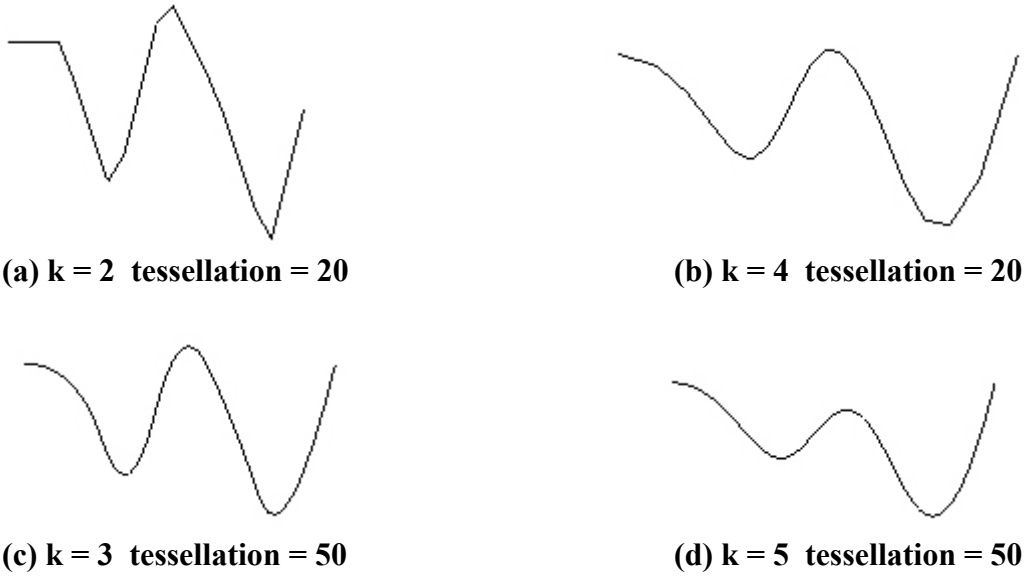
```
NurbsCurve : X3DParametricGeometryNode {
MFVec3f  [in,out] controlPoint [] (-∞ ,∞ )
SFInt32  [in,out] tessellation 0  (-∞ ,∞ )
MFDouble [in,out] weight       [] (0,∞ )
MFDouble []       knot         [] (-∞ ,∞ )
SFInt32  []       order        3  [2,∞ )
}
```

Figure 7.22 NurbsCurve Specification (From Ref. [11].)

By convention, variables given the "[]" qualifier are static values that must be specified upon initialization, with accessType "initializeOnly." If no weight values are given, the default non-rational vector of all ones will be created. Similarly, if no knot vector is given, the standard open uniform knot vector is automatically created at runtime. The curve defaults to an order of 3, or degree 2. In most cases, acceptable results may be obtained with an order of 4, or cubic degree. Looking back at Equation 7.2c, the calculation of basis functions for NURBS are very mathematically intensive, as well as recursive. Increasing the degree of the curve adds an extra level of recursion to this process. Therefore, it is ideal to use the smallest possible order to obtain the desired curve smoothness.

Variables that can be animated include the controlPoint vector, its weightings, as well as the tessellation of the curve. Once again, if Equation 7.4a is analyzed, notice that the control points, represented by the B_i vector, are only applied to the curve after the basis functions have been calculated. Since the basis functions rely solely on the degree of the curve and the knot vector, dynamically changing the control points of a curve

requires only that the top-level summation be recomputed. This allows for real-time animation of the NurbsCurve node. Figure 7.23 shows several examples of the NurbsCurve node in practice. These examples illustrate the effect of using curves of varying degree and tessellation. Tessellation is accomplished by computing Equation 7.4 for a given number of points along the knot vector between the first and last value. The default tessellation of 20 means that the first, last, and 18 intermediate points are calculated along the curve. These 20 points form the basis of 19 line segments that approximate the curve using an IndexedLineSet node within X3D. Simply put, the higher the tessellation, the smoother the curve will appear on screen, however the more computationally expensive the curve becomes.



(a) k = 2 tessellation = 20 **(b) k = 4 tessellation = 20**

(c) k = 3 tessellation = 50 **(d) k = 5 tessellation = 50**

Figure 7.23 NurbsCurve Node Examples.

2. Nurbs Surface Prototype

The specification for the NurbsSurface node is shown in Figure 7.24.

```
NurbsSurface : X3DParametricGeometryNode {
  MFVec3f  [in,out] controlPoint  []   (-∞ ,∞ )
  SFNode   [in,out] texCoord      []
  SFInt32  [in,out] uTessellation 0    (-∞ ,∞ )
  SFInt32  [in,out] vTessellation 0    (-∞ ,∞ )
  MFDouble [in,out] weight        []   (0,∞ )
  SFBool   []       ccw           TRUE
  SFBool   []       solid         TRUE
  SFInt32  []       uDimension    0    [0,∞ )
  MFDouble []       uKnot         []   (-∞ ,∞ )
  SFInt32  []       uOrder        3    [2,∞ )
  SFInt32  []       vDimension    0    [0,∞ )
  MFDouble []       vKnot         []   (-∞ ,∞ )
  SFInt32  []       vOrder        3    [2,∞ )
  SFBool   []       closed        FALSE
}
```

Figure 7.24 NurbsSurface Specification(From Ref. [11].)

Notice how many variables are now qualified with the surface direction u or v. Similar to NurbsCurve, default values for the weight, order, and knot vectors will be created if not specified by the user at runtime. ControlPoint and tessellation values can again be animated. However, in contrast to a NurbsCurve, the computational expense to compute a NurbsSurface has increased to $N*M$ where N and M represent the u order and v order of the curve, respectively. Changing the controlPoints vector still only requires that the final, now double, summation be recomputed.

Figure 7.25 shows a simple NurbsSurface, describing a plane. The dimension in each direction is 5, for a total of 25 control points. Computing the double summation for this surface requires on the order of 25 multiplications and 25 sums for each tessellation in each direction for a total of 10,000 multiplications and 10,000 summations with a default tessellation of 20 in each direction. This task is not too difficult for most modern processors.

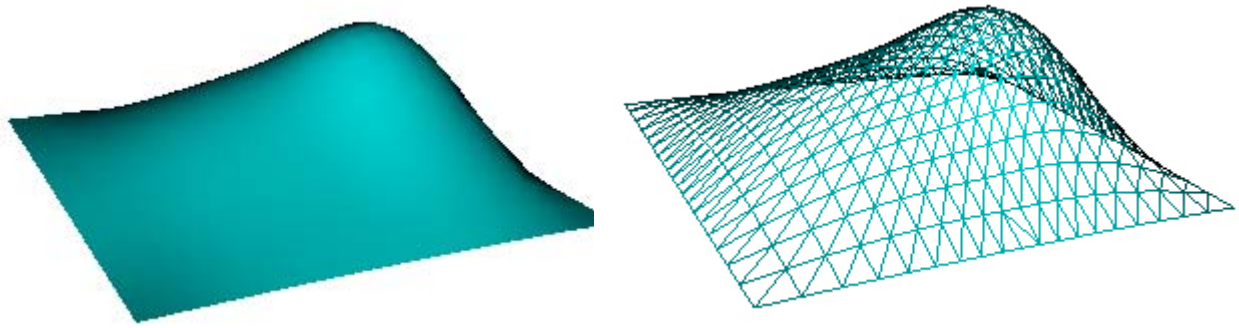
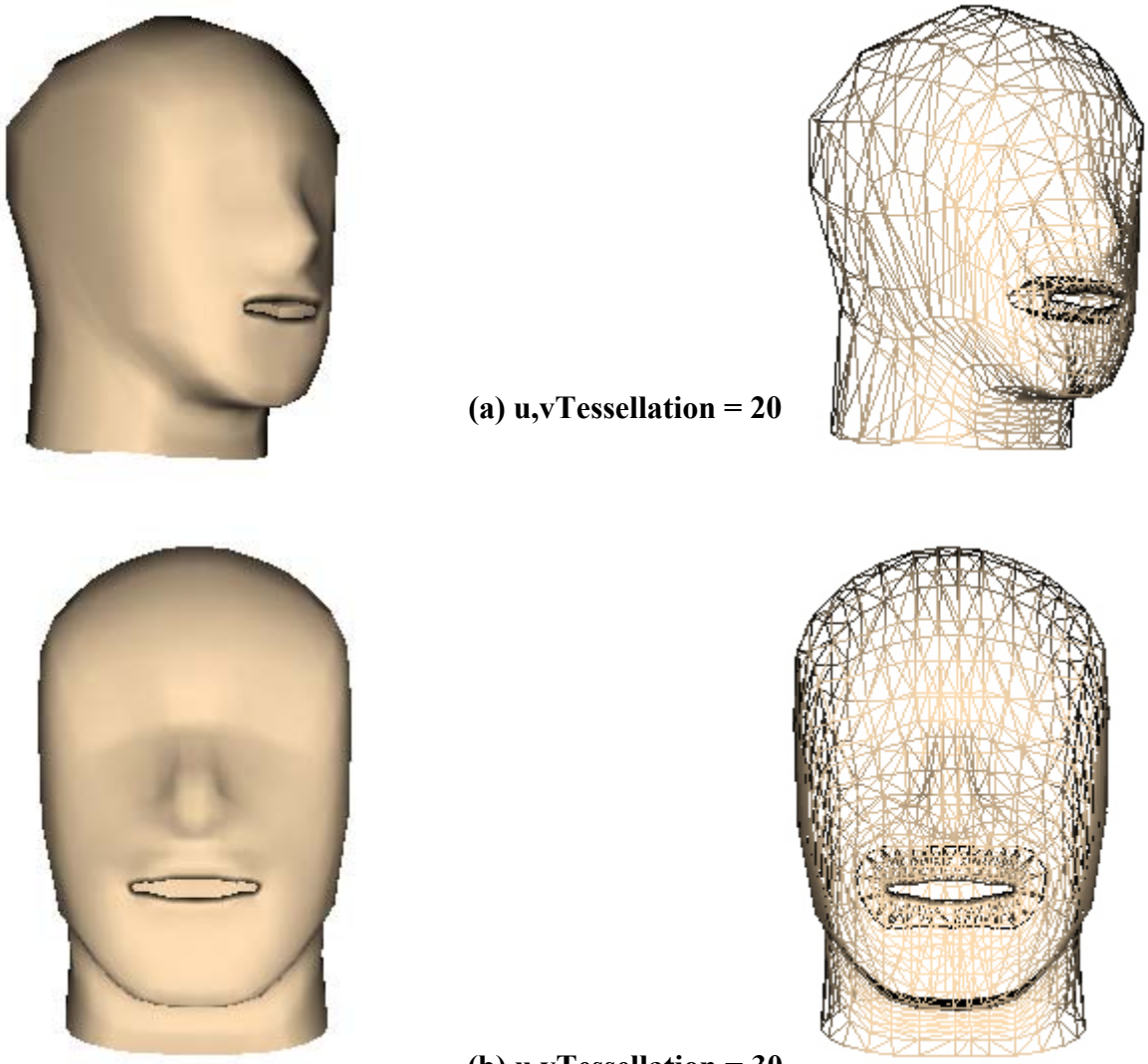


Figure 7.25 NurbsSurface Plane.

However, consider the NurbsSurface head shown in Figure 7.26. This surface is created by 24 points in the u direction and 30 points in the v direction for a total of 720 control points. The final summation in this case requires on the order of 720 multiplications and 720 sums for each of 400 tessellation points or nearly 576,000 total floating point computations. If the entire summation is recomputed for the smallest change in one control point, then even the fastest computers available would have difficulty doing such a large number of floating point operations for each frame at a rate of 30 frames per second. Not only this, but there is considerable overhead involved in using the Java interface to X3D, further slowing down the frame rate.



(a) u,vTessellation = 20

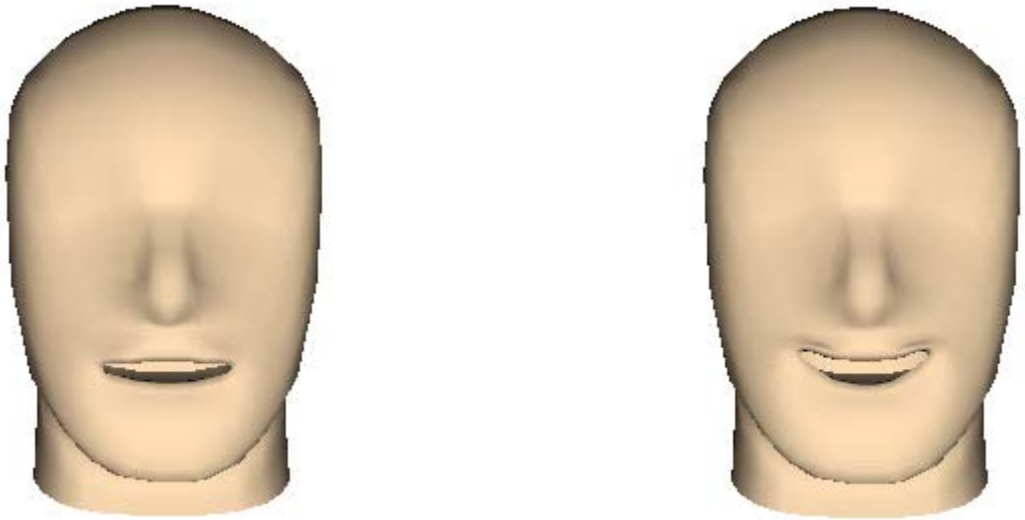
(b) u,vTessellation = 30

Figure 7.26 NurbsSurface Head Tessellation Example.

Remembering the first principles of B-Splines, however, quickly provides a solution to this problem. When one control point moves, the entire surface does not change. Instead, the change is localized around the control point that moves. Thus there must be a way then to only compute the surface around points that change. Not only does the degree of the curve represent the smoothness of the surface, it also defines the extent to which a control point exerts influence on the surrounding surface. In general, the order of the curve determines how many neighboring control points are affected by the changes

in one control point. Even though the NurbsSurface Head has 720 control points, the order in any one direction is only 4. This means that the total surface might only change by about 1% for movement in a single control point.

Moreover, many NURBS surfaces do not require that all control points be animated at once. Considering the NurbsSurface Head again in Figure 7.27, to animate the mouth, less than ten control points must be changed. By incrementally updating the surface based only upon the control points that change, a 1600% increase in computational efficiency is reached. Examples for the animated head have shown improvements of this much and more after incremental animation algorithms are implemented. What once ran at a dismal 0.2 frames per second now reaches as high as 15 frames per second.



(a) Pose One **(b) Pose Two**

Figure 7.27 NurbsSurface Head Animated.

A final word on the tessellation of NURBS surfaces is in order. Despite the fact that massive improvements in speed can be obtained simply by incrementally updating the surface around points that change, unfortunately tessellation algorithms are not that lucky. Since tessellation requires increasing the number of points along the knot vector that are computed, a change in tessellation means new values for the basis functions at

those points must also be computed. It is thus best to determine the necessary tessellation value ahead of time and to stick with it. Level of detail can be accomplished through existing nodes within X3D that allow for several versions of the same shape to be used for varying resolution requirements. In this way, a NURBS object is only needed at the closest distance. Fast polygonal shapes may be used for lower resolution estimates.

3. Nurbs Position Interpolator Prototype

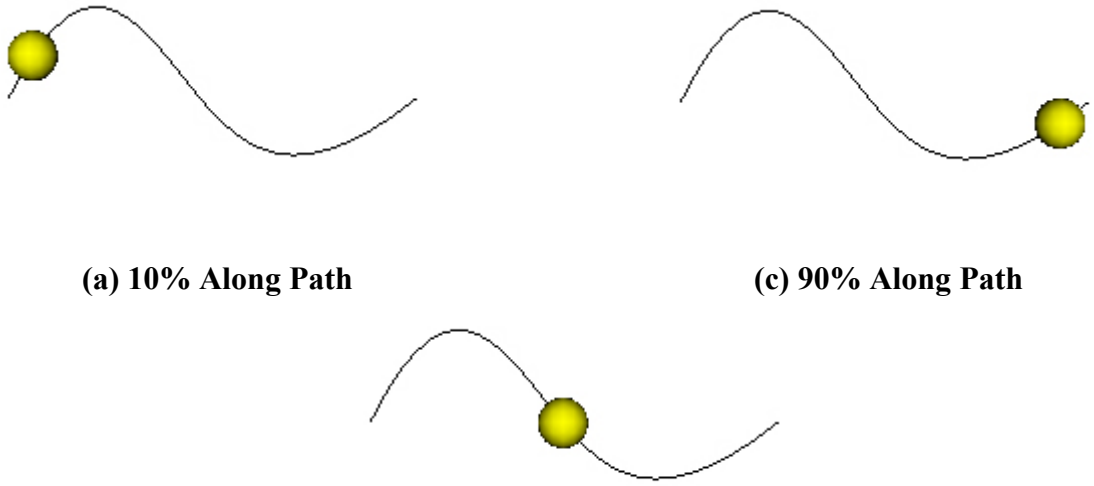
The specification for the NurbsPositionInterpolator node is shown in Figure 7.28.

```
NurbsPositionInterpolator : X3DInterpolatorNode {
  SFFloat  [in]     set_fraction     (-∞ ,∞ )
  SFBool   [in,out] fractionAbsolute TRUE
  MFDouble [in,out] key              [](-∞ ,∞ )
  MFVec3f  [in,out] keyValue         [](-∞ ,∞ )
  MFDouble [in,out] weight           [] (-∞ ,∞ )
  MFDouble [in,out] knot             [] (-∞ ,∞ )
  SFInt32  [in,out] order             3  (2,∞ )
  SFVec3f  [out]    value_changed
}
```

Figure 7.28 NurbsPositionInterpolator Specification(From Ref. [11].)

The true power of a NurbsCurve is the ability to animate an object along the curve as an animation path. This is the goal of the NurbsPositionInterpolator node. Many of the fields are the same as for the NurbsCurve. However, there are no control points specified. Instead, control vertices are found in the keyValue field. In accordance with traditional X3D PositionInterpolator concepts, the “key” field defines time points at which the value in the keyValue field will be reached. The control points are simply goals along the way, and the keys are the times, one corresponding to each control point, when each goal is reached. This brings up an interesting point. Unlike in traditional linear position interpolators where the path is guaranteed to travel through each keyValue, the very nature of NURBS curves are such that the only control points that the path is guaranteed to touch are the first and last. Therefore, care must be taken to ensure that the total shape of the path is considered so that the correct motion is obtained. For this reason, the NurbsPositionInterpolator allows the user to see the path that has been

created so that it may be adjusted as needed to meet the specified purpose. Figure 7.29 shows an example of using the NurbsPositionInterpolator for a sphere traversing a NurbsCurve.



(a) 10% Along Path **(c) 90% Along Path**

(b) 50% Along Path

Figure 7.29 NurbsPositionInterpolator Example.

4. Future Nodes

The world of NURBS is not limited to the three nodes discussed in this chapter, but instead spans volumes on specific techniques used to create and manipulate complex objects. One additional node that would add greater flexibility to the current X3D implementation of NURBS is a NurbsOrientationInterpolator. Recently added to the NURBS specification for X3D, the goal of this node is not to follow the position of a NURBS curve, but as the name suggests, the orientation of the curve. Through calculation of the normal at each point along the curve, it is possible to output an orientation that may be routed to the orientation of the object following the path. This node is especially useful for computing the orientation of flight paths, releasing the animator from worrying about the direction of the plane while it follows the path. In addition to following the orientation of the path, additional work could allow for a certain degree of banking around turns and deviations from the normal based on the curvature of the path.

Although NURBS reflect smooth mathematical descriptions of complex objects, the general form of a NURBS surface requires it to be continuous across both dimensions. Many researchers have attempted to overcome this limitation through the implementation of trimmed NURBS. As the name suggests, a trimmed NURBS surface is one in which a specified patch has been trimmed out, or removed. Figure 7.30 shows an example of trimmed NURBS, implemented in the graphics package Maya™. Much of the difficulty in trimming surfaces lies in maintaining boundary conditions along multiple surface curves, or isoparams.

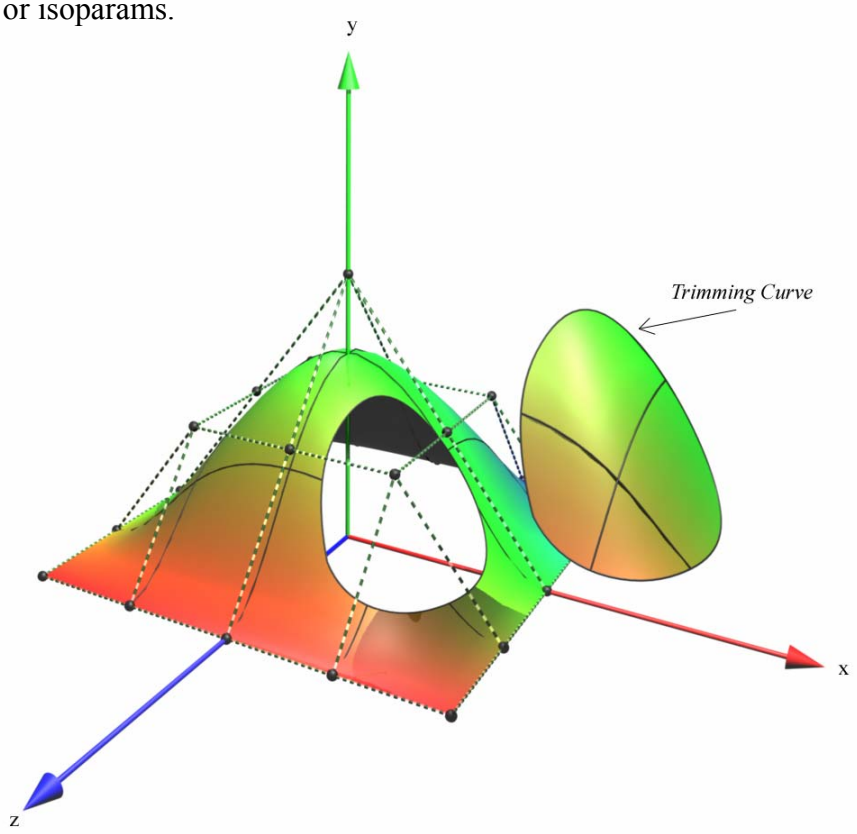


Figure 7.30 Trimmed NURBS Example.

Both NurbsOrientationInterpolator and TrimmedNurbs nodes have been added to the X3D specification. With time, corresponding implementation of these nodes will help bring greater flexibility to the creation of complex objects within X3D. However, significant flexibility to import NURBS into X3D from commercial software packages already exists as the next section explains.

5. Maya™ Authoring Tool Plug-in for X3D Export

Much of the inspiration to implement the NURBS specification for X3D came from experience working with the popular graphics authoring package Maya™. Created by Alias|Wavefront, Maya™ represents the current industry standard in 3D modeling and animation. One of the earliest topics presented when learning a 3D modeling program, the power of NURBS is quickly discovered as a fast and straightforward method to create models of incredible complexity and smoothness. The properties of NURBS, previously mentioned, make them the perfect sculpting medium for artists familiar with clay molding. Pushing and pulling on individual vertices translates to smooth and efficient modifications to the "clay."

While Maya represents an amazingly powerful tool by which NURBS can be created and molded, there was no available exporter to translate these objects into the X3D format. This is due mainly to the fact that until now there has not been a general implementation of NURBS within X3D. Since X3D by its very nature is not a modeling tool but a medium wherein to develop interactive worlds, implementing NURBS within X3D without a method to import them invalidates the original effort. When the specification for VRML was released in 1997, a commercial plug-in for Maya™ to export polygonal objects was also soon released. This plug-in does export both polygon and NURBS components. However, NURBS are first converted into polygonal objects and lose all previous properties of continuity embedded in the NURBS mathematical model. Since NURBS now have a general implemented in X3D, there must be a way to export them in their original structure from Maya™ as well.

This is the goal of the X3DNurbsExport plug-in for Maya™, created as part of the general NURBS implementation. The plug-in is very simple and is in its early stages of development. As such there are many quirks yet to be worked out. However, it is the first fully functioning and distributable plug-in for Maya™ that is able to export NURBS geometry in its true sense, directly to the current X3D format. This, above all else, gives credibility to the NURBS implementation as a meaningful contribution to the X3D community. The fact that most of the examples presented thus far were originally created in Maya™, and exported to X3D intact, validates both the implementation model as well

as the plug-in. Figure 7.31 presents two examples of more complex objects created within Maya™ and exported to X3D. These models each contain a combination of six or more NURBS surfaces.

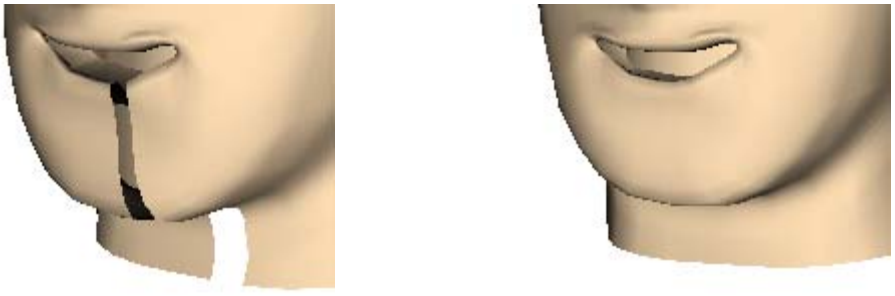
(a) Fred the Bunny **(b) Hurricane Lantern**

Figure 7.31 NurbsSurfaces Exported by Maya X3DnurbsExporter Plug-in.

The X3DNurbsExport Plug-in is written within the framework of Microsoft Visual C++ and the plug-in wizard provided by Alias|Wavefront. In its most basic form, it reaches into the geometry dependency graphs of Maya's internal data structure and pulls out the necessary fields to create a NurbsSurface node in X3D. The controlPoint, weight, knot, and order fields are pulled directly from Maya and rewritten to match the X3D implemented format. One initial problem that surfaced during the creation of the plug-in is that closed surfaces, those in which the first control point align with the last control point, did not render properly. Figure 7.32 shows an example of this.

Figure 7.32 Incorrect NURBS Surface Closure.

As it turns out, there are many different ways to specify the end conditions of a NURBS surface. Maya™ chooses to use what is commonly referred to as pseudovertices. These are extra control points that extend beyond the drawn surface. They are included to compute the surface, but not actually drawn out. As there are various implementations of pseudovertices, unless the specific nature of these extra vertices is known, it is very difficult to translate NURBS surfaces created in this manner. For this reason, an extra field is added to the NurbsSurface node in order for the user to specify whether or not the surface is closed. By rebuilding these closed surfaces in Maya™ to adhere to more traditional NURBS creation methods, and by setting the "closed" field to "true" in the NurbsSurface node, the resultant surfaces will be rendered properly within X3D. Figure 7.33 shows the result of rebuilding the NURBS head within Maya™, as well as the effect the closed field has upon the surface. Although the closed option specifically tells the algorithm to connect the first and last control points when tessellating, no seams are evident upon visual inspection of the resulting object.



(a) Closed Field False **(b) Closed Field True**

Figure 7.33 Rebuilt NURBS Surfaces for End Conditions

E. SUMMARY

This chapter presents the tools necessary to realize the visual interpretation of 3D ISAR. The X3D specification, in conjunction with X3D-Edit and Java, provides a powerful method of bringing interactive virtual content to web-based applications. This software is constantly being improved and updated due to its close ties to NPS. As such, it makes an excellent platform for developing 3D ISAR simulations at NPS. The general 3D ISAR concept is presented through the overlay of sample 2D images on a 3D ship model. Through the correlation of 2D range-Doppler profiles, 3D geometry can be extracted. The next logical step in the continuation of research into this topic is to apply the equations developed in Chapter VI to an X3D simulation. Within this realm of virtual reality, the full capability of NURBS may be employed to generate complex, high-resolution graphical models to correlate ship geometry with ISAR data. Eventually, methods of more closely connecting the DIS and X3D simulations can be explored to provide seamless interaction between both platforms.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The ability to discern the enemy amongst the battlefield becomes increasingly difficult as the battlefield landscape changes to reflect both massive oceans and crowded littoral passageways. To detect a vessel provides little useful information if the target cannot be classified. Furthermore, modern weapon systems dictate that surveillance aircraft remain as far from the target as possible to maintain a safe region of operation. Providing both detection and classification capability at full range for potential threats, Inverse Synthetic Aperture Radar gives an operational advantage to those deploying it. However, this technology is available not only to the United States and allied countries, but also to its potential enemies. It is therefore necessary to develop counter ISAR methods. The Digital Image Synthesizer is a key step forward in realizing this goal.

Validating the DIS is done on several levels, from software simulation to full hardware implementation. Extending the software validation began by Fernando Le Dantec in [1], the integrity of hardware design choices is scrutinized through the Adder Overflow Analysis in Chapter IV. Specifically, it was shown that the 16-bit summation adder at the end of the DIS is susceptible to detrimental overflows when false-targets of realistic size are simulated. Due to the binary nature of the power-of-two gain shifter, however, a viable, low-cost solution to this problem was presented which allows for the total signal gain to be reduced as needed to bound the output signal to within reasonable levels.

The groundwork for further DIS validation is laid through the creation of a software model, described in Chapter V, which is capable of processing realistic ISAR simulation data. This model is able to both display the standard 2D ISAR images, and run the software DIS model with coefficients pulled from the acquired data. Comparison of the results provides verification of the algorithms behind the software model and, thus, also behind the hardware implementation on which the software is based. Once fabricated, the true hardware DIS system can be run from these extracted coefficients to validate the integrated circuit chip as well.

Traveling beyond the current DIS limits of 2D ISAR, this thesis provides the foundation for research into the development of three-dimensional ISAR and its application to the DIS. Through a coherent examination of current attempts at 3D ISAR, a valid direction in which to steer future research can be found. Specifically, promising results have been produced through the analysis of the backscattering effects in the phases of adjacent range cells from current 2D ISAR images when processed through multiple receivers. The added dimension of elevation to ISAR images allows for enhanced feature recognition in addition to standard target classification. This can be especially useful in the littoral combat region where the ability to quickly detect and classify small, fast-moving craft is of the utmost importance.

Through the development of tools such as X3D and its NURBS subsystem, a specific approach for analyzing 2D ISAR images in light of 3D techniques was presented. The overlaying of multiple radar images on a geometric ship model, followed by geometric feature extraction, allows the user to fully explore the correlation of image data to physical geometry. This correlation can uncover novel feature recognition schemes as well as previously unnoticed weaknesses in current imaging techniques. By building on the foundation laid through this thesis and its corresponding software implementations, the next step forward in both ISAR countermeasures and 3D ISAR image generation may be realized.

B. RECOMMENDATIONS FOR FUTURE WORK

The research conducted on the feasibility of 3D ISAR represents only the tip of the information iceberg on 3D radar imaging concepts. Building upon the foundation of 3D interferometric ISAR and 3-receiver ISAR, the algorithms for these techniques can be implemented and tested based on actual multi-receiver experiments. Going one step further, the 2D ISAR images created by these approaches may be inserted into an X3D virtual simulation as described in Chapter VII. From there, it is possible that new methods of image formation and correlation may be discovered to further enhance the algorithms. By developing a practical method of visualizing 3D ISAR, radar operators can more effectively accomplish their mission.

The DIS is nearing completion and may be fabricated by 2004. It is thus important to test the actual chip once in production. While traditional testing is sure to be done to verify its operation, additional tests can presently be developed by which to explore the current design capabilities of the DIS with respect to 3D ISAR. If, by the time the actual chip is fabricated, a working software 3D DIS model is completed, this additional testing is simply the next step in the natural progression of the development process and an excellent opportunity for testing current algorithms. Regardless of the timeframe, however, future work on either the algorithms behind 3D ISAR, or new methods of visualizing the resultant images, will be helpful in both developing the next generation of high-resolution radar devices and in supporting future military operational research.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

The appendix contains a listing of acronyms mentioned in the thesis, as well as the source code for the modifications to the DIS, and distribution information on how to obtain an accompanying cd-rom.

A. ACRONYMS

2D	Two-Dimensional
3D	Three-Dimensional
DIS	Digital Image Synthesizer
DRFM	Digital Radio Frequency Memory
IC	Integrated Circuit
ISAR	Inverse Synthetic Aperture Radar
MATLAB	Mathematical Laboratory
SAR	Synthetic Aperture Radar
X3D	Extensible 3D

B. DIS SOFTWARE UPDATE

```
%************************************************************************
%  BeginDataProcessing.m                                                *
%                                                                       *
%  - Front end to DIS simulation.                                       *
%  - Creates 2D image based on input data set                           *
%  - Extracts coefficients and runs DIS with those coefficents          *
%  Modified by C. Adams June 2003 based on previous software from       *
%  F. Le Dantec and P.E.Pace                                            *
%                                                                       *
%************************************************************************

M = 32; % Number of returns per pulse = number of range bins
windowSize = 128; % Number of pulses within window
N = 128; % Total number of pulses in data set
bw = 500E6; % Bandwidth
tau = 1/bw; % Pulse width
fs = 2*bw; %Sampling Frequency
prf = 200; %Pulse Repetition Frequency
pri = 1/prf; %Pulse Repetition Interval
```

```
k = bw/tau;                  % Pulse compression rate (deltaBW/uncompr.pWidth)
nbitsdop = 5;                % # of bits used for phase quantization
Ts    = 1/fs;                % Sampling time
t0    = Ts:Ts:tau;          % Interpulse sample index
rag_pts = length(t0);     % Number of samples per chirp pulse (range cells)
rg_pts = max(rag_pts,M*10); % # of range FFT pts including padding

fullDataSet = zeros(M,N); % <-----   Data Set goes here

%For each window set of data, copy windowSize pulses from fullDataSet
% into summedReturnI
for winNum = 1:N-windowSize+1
    summedReturnI = zeros(M,windowSize);
    counter = 1;
    for index1 = winNum:windowSize+winNum-1
        for index2 = 1:M
            summedReturnI(index2,counter) = fullDataSet(index2,index1);
        end
        counter = counter + 1;
    end

    % Create Reference Signal for Cross-corellation
    referenceS = zeros(M,windowSize);
    for n = 1:windowSize
        for m = 1:M
            t = ((m-1)/fs + (n-1)*pri);
            referenceS(m,n) = rect(t,tau)*exp(j*(pi*bw/tau)*(m/fs+n*pri)^2);
        end
    end

    % Range Compression
    I_kn = fft(summedReturnI);
    S_kn = fft(referenceS);
    complexRangeProfile = ifft((I_kn .* conj(S_kn)));

    %load targetSum;
    %complexRangeProfile = targetSum;

    % Azimuth Compression
    originalImage = flipud(abs(fft(complexRangeProfile.')));

    % ISAR IMAGE PLOTS (Range Doppler profile):
    figure;
    dopp = (1:size(originalImage,1))*prf/windowSize; % Doppler axis
    rang = 1:size(originalImage,2);                    % Range axis
    colordef black                                  % Background color
    colormap(hot(100))
    surf(rang,dopp,originalImage), shading interp, lighting phong, view(0,90);
    title('Range-Doppler profile of true target');
    xlabel('Down Range Cells');
    ylabel('Doppler frequency [Hertz]');
    %axis([0 32 0 100])
    axis tight

    % Extract Coefficients and Run DIS
    extractCoefficients(complexRangeProfile, M, windowSize, Ts, tau, nbitsdop,
t0, rag_pts, rg_pts, winNum)
    runDIS(windowSize,rg_pts,M,rag_pts,k,t0,prf,nbitsdop,winNum)

    % Go get the next window's worth of data
end

%*************************************************************************
```

```
%  ExtractCoefficents.m                                                    *
%                                                                          *
%  - Extracts coefficients and stores them in paraMULTIq4Ship_winNum.txt   *
%  for the window number winNum                                            *
%  Modified by C. Adams June 2003 based on previous software from          *
%  F. Le Dantec and P.E.Pace                                               *
%                                                                          *
%**************************************************************************

function extractCoefficients(complexRangeProfile, M, windowSize, Ts, tau,
nbitsdop, t0, rag_pts, rg_pts, winNum)

% CREATION OF GAIN MODULATION COEFFICIENTS :

amp = abs(complexRangeProfile);          % Magnitude of combined signal.
gain = ones(M,windowSize);             % Default gain coefficient is 1.
fq = 1.5;
for i1 = 1:M,
    for i2 = 1:windowSize,
        normAmp = amp(i1,i2)/max(max(amp)); % Magnitudes are normalized.
        for s=1:10,
            if normAmp > (2^-(11-s))*fq, % Thresh: 0.0015....,0.375,0.75
                gain(i1,i2) = 2^s;       % Gain:   2,4,8,16,......512,1024
            end                          % Assigns 11 power-of-two gains.
        end
    end
end

% CREATION OF PHASE MODULATION COEFFICIENTS :

vphase1  = angle(complexRangeProfile);            % Phase of combined signal
vphase2  = vphase1*2^nbitsdop/(2*pi);   % Adjusts values from 0 to 31.99
phaseinc = zeros(M,windowSize);
for i6 = 1:M,                             % Turn phases into phase increments
  for i7 = 1:windowSize,
      if i7==1,
         phaseinc(i6,i7) = phaseinc(i6,i7)+vphase2(i6,i7);
      else                                % Finds phase increments bet/pulses
       phaseinc(i6,i7) = phaseinc(i6,i7-1)+vphase2(i6,i7-1)-vphase2(i6,i7);
      end
  end
end
for i8 = 1:windowSize,
   for i9 = 1:M,                  % 5 bit quantization in 32 levels.
      phasecoeff(i9,i8) = fix(rem(phaseinc(i9,i8)+32,32));
   end
end

%    MODULATION COEFFICIENTS ARE STORED SERIALLY, SO THEY CAN BE LOADED BY
% THE DIS SIMULATION PROGRAM :

filename = ['paraMULTIq4Ship_',int2str(winNum),'.txt'];
f4 = fopen(filename,'w');  % Stores modulation coefficients
fprintf(f4,'%d\r\n',rag_pts);                  % # of chirp pulse samples
fprintf(f4,'%d\r\n',windowSize);               % # of Doppler bins
fprintf(f4,'%d\r\n',M);                % # of taplines (target extent)
for aa=1:windowSize,
   for bb=1:M,
      fprintf(f4,'%d\r\n',gain(bb,aa)); % Stores gain coefficients
   end
end
for aa = 1:windowSize,
   for bb = 1:M,
```

```
      fprintf(f4,'%d\r\n',phasecoeff(bb,aa));  % Stores phase coefficients
   end
end
fclose(f4);

% PLOT OF THE GAIN QUANTIZATION SCHEME :

figure
plot(amp/max(max(amp)),gain,'b*');       % Plots gain coefficient distrib.
title('Plot of the Gain Quantization Scheme');

%************************************************************************
%  RunDIS.m                                                             *
%  - Modified version of Mathost_v6.m                                   *
%  - Creates and stores an intercepted chirp pulse as generated by a    *
%  phase sampling DRFM.                                                 *
%  - Creates the reference signal required for ISAR range compression.  *
%  - Runs 'simhwchk_v5.m' to execute the DIS and ISAR simulation        *
%  - 'image_analysis.m' can be run if infinite resolution case exists.  *
%                                                                       *
%  Modified by C. Adams June 2003 based on previous version from P.E.Pace *
%  and F. LeDantec                                                      *
%************************************************************************

function runDIS(windowSize,rg_pts,M,rag_pts,k,t0,prf,nbitsdop, winNum)

% GENERATION OF DRFM SIGNAL CONTAINING PHASE SAMPLES FOR ALL PULSES :
pri_rg_phaseq = zeros(windowSize,rag_pts);  % Initializes DRFM phase matrix
p = 2*pi/(2^nbitsdop);                  % Phase quantization factor
pri = 1/prf;
fc= (9:0.5:12.5)*1e9;
for i=1:log2(windowSize/8);
    fc=[fc fc];
end;
for idx1 = 1:windowSize,                    % loop per pulse
   t1 = t0 + idx1*pri;                  % PRI time shift for pulse samples
   oldphase = 2*pi*(k*t1.*t1)/2;        % Chirp pulse phase samples
   oldphase = rem(oldphase,2*pi);       % limit phase from 0 to 2pi
   pri_rg_phaseq(idx1,:)=fix(oldphase/p);%Phase quantized in 2^nbit levels
   crefc(:,idx1) = exp(j*oldphase.');   % Store phase for ISAR reference
end

% STORAGE OF DRFM PHASE MATRIX :
f2 = fopen('rawint.txt','w');           % Opens file for DRFM ph. storage
for i = 1:windowSize,                       % Loop for all pulses.
   int_raw = pri_rg_phaseq(i,1:rag_pts-1);
   fprintf(f2,'%d,',int_raw);           % Stores pulse phase samples
   int_raw = pri_rg_phaseq(i,rag_pts);
   fprintf(f2,'%d\r\n',int_raw);        % Stores last pulse phase sample
end;
fclose(f2);

% GENERATION OF ISAR REFERENCE FOR RANGE COMPRESSION (FAST CORRELATION) :
cref = conj(fft(crefc,2*rg_pts-1));    % FFT of ISAR reference
save pc_ref cref t1                    % Saves FFT of ISAR ref.

% DIS SIMULATION, IMAGE GENERATION AN ANALYSIS :
simhwchk_v5(rg_pts, prf, windowSize, M, winNum);                          %
DIS simulation & ISAR procesing
%Image_Analysis;

%************************************************************************
%  simhwchk_v5.m                                                        *
```

```
%                                                                       *
%  Simulates the DIS architecture by:                                   *
%  - Loading its inputs from files 'paraMULTIq4Ship.txt' and 'rawint.txt' *
%  - Processing the modulations of the tap array.                       *
%  - Generating the output chirp.                                       *
%  Simulates the ISAR processing and plots the range-Doppler profile of   *
%  the false target as seen in the radar screen.                        *
%                                                                       *
%  Modified by C. Adams June 2003 based on previous version from P.E.Pace *
%  and F. LeDantec                                                      *
%***********************************************************************

function simhwchk_v5(rg_pts, prf, windowSize, M, winNum)

filename = ['paraMULTIq4Ship_',int2str(winNum),'.txt'];
fid = fopen(filename,'r');     % Opens file w/DIS coefficients
tmp = fscanf(fid,'%f');                    % Loads file contents
nRangeCell = tmp(1);                       % number of pulse samples
nDopplerCell = tmp(2);                     % number of pulses integrated
M = tmp(3);                             % number of DIS range taps
gainIn = tmp(4:4+M*nDopplerCell-1);   % Extracts gain coefficients
gain = reshape(gainIn,M,nDopplerCell);% Arranges them in a matrix
gain = gain';                              % Transpose
phi = tmp(4+M*nDopplerCell:end);      % Extracts gain coefficients
phasInc = reshape(phi,M,nDopplerCell);% Arranges them in a matrix
fclose(fid);
raw = zeros(nDopplerCell,nRangeCell);       % Initializes DRFM phase matrix
fid = fopen('rawint.txt','r');              % Opens file w/DRFM samples
for r = 1:nDopplerCell,                     % Loop per pulse
    for i2 = 1:nRangeCell-1,                % Loop per sample
        raw(r,i2)=fscanf(fid,'%f',1);       % Loads DRFM phase sample
        comma = fscanf(fid,'%c',1);         % Loads comma character
    end
    raw(r,nRangeCell) = fscanf(fid,'%f',1); % Last pulse phase sample
end
fclose(fid);
[row,col] = size(raw);
phaDRFM = [raw,zeros(row,M-1)];       % Trailing zeros to clear taps

colordef white
set(0,'defaultAxesFontSize',12);            % Font size for graph labels
figure                                      % Plots one intercepted pulse
plot( real( exp(j*phaDRFM( 1,1:(nRangeCell+M-1) )/32*2*pi )),'b')
title('Plot of an intercepted ISAR chirp as generated by a DRFM');

depthLUT = 32;                              % Size of the LUT table
load -ascii cosine8.txt                     % Loads LUT tables
load -ascii sine8.txt
lutOut  = zeros(M,1);                  % Initializes LUT output
tapOut  = zeros(nRangeCell + (M-1),M);
o_res   = 0.25;                             % Minimum output bit resolution
overflow= 0;                                % Initializes overflow counter
peak    = 0;                                % Initializes maximum amplitude
limit   = 8192;                             % Adder overflow limit 2^13
partial_tapsum = zeros(nRangeCell+M-1,1);

for batchCnt = 1:nDopplerCell,                        % Loop per ISAR pulse
    disp(['Processing Pulse ',num2str(batchCnt)]);
    for inPlsCnt = 1:(nRangeCell+M-1),          % Per out plse sample
        tapIn(1:M) = phaDRFM(batchCnt,inPlsCnt);% Loads DRFM Pha in//
        phAddOut= tapIn(1:M)'+phasInc(:,batchCnt); % Pha adder output
        tmp = rem(phAddOut+depthLUT,depthLUT)+1;      % Phas limit bet 0-31
        lutOut = cosine8(tmp) + j*sine8(tmp);         % LUT table output
```

```
        gainOut = gain(batchCnt,:)'.*lutOut;          % Gain modulation
        if inPlsCnt > nRangeCell,                     % Shuts down taps
            gain0ut(1:(inPlsCnt-nRangeCell)) = 0;     % after pulse passes
        end
        for idx3=1:M,
            delay = idx3-1;           % Out.adder register delay bet/taps
            tapOut(inPlsCnt+delay,idx3)=fix(gainOut(idx3)/o_res)*o_res;
        end                           % Applies minimum register resolution
        sum1=0;
        for tt=min(inPlsCnt,M):-1:max(1,inPlsCnt-nRangeCell),
           sum1 = sum1 + tapOut(inPlsCnt,tt);
           Iout=real(sum1);           % Overflow simulation using +/-limit
           Qout=imag(sum1);
 %          if Iout >= limit,                    % Positive oflow (in-phase)
              %Iout = limit;
%               sum1 = Iout-2*limit + j*Qout;    % Adder output w/overflow
%               overflow = overflow + 1;         % Increases counter
%           elseif Iout < -limit,                % Negative oflow (quad-pha)
              %Iout = -limit;
%               sum1 = Iout+2*limit + j*Qout;    % Adder output w/overflow
%               overflow = overflow + 1;         % Increases counter
%           end
%           if Qout >= limit,                    % Positive oflow (in-phase)
              %Qout = limit;
%               sum1 = Iout + j*(Qout-2*limit);  % Adder output w/overflow
%               overflow = overflow + 1;         % Increases counter
%           elseif Qout < -limit,                % Negative oflow (quad-pha)
              %Qout = -limit;
%               sum1 = Iout + j*(Qout+2*limit);  % Adder output w/overflow
%               overflow = overflow + 1;         % Increases counter
%           end
           peak=max(max(abs(Iout),abs(Qout)),peak);   % End of oflow simul.
           partial_tapsum(inPlsCnt,1) = sum1;
        end
    end
    finalAdderOut(batchCnt,:)=partial_tapsum.'; % DIS output mod. chirps
end
overflow                                        % Number of DIS overflows
peak                                            % Maximum DIS amplitude
dac_res = 0.25;                                 % DAC resol 0.25 for 16bit
finalAdderOut = fix(finalAdderOut/dac_res)*dac_res;
figure                                %  Plots 1 DIS output chirp pulse
plot( real( finalAdderOut(1,:) ),'b'),axis tight
title('Plot of the DIS output chirp pulse');


% ISAR PROCESSING :

load pc_ref                           % Loads FFT of ISAR reference
priRgMapShift = zeros(M/8*9,nDopplerCell);
    % Range compression:
pcRefMapShift = fft(finalAdderOut.',2*rg_pts-1).'; % FFT of DIS output
for idx = 1:nDopplerCell,                       % Loop for each pulse
  tmp  = cref(:,idx).*pcRefMapShift(idx,:).';   % Fast correl(FFT product)
  tmp2 = ifft(tmp);                             % Obtains range profile
  priRgMapShift(:,idx)=tmp2(1:fix(M/8*9));% Collects range profiles
end
    % Azimuth compression:
    % FFT of matrix of range profiles in the pulse index dimension (time)
    % gives the target range Doppler profile.
dpRgMapShiftMOD = abs(fft(priRgMapShift.',nDopplerCell));
quantimage = dpRgMapShiftMOD(1:nDopplerCell*100/prf,1:fix(M/8*9));
save finresoldata quantimage            % Saves Range Doppler profile
```

```
% ISAR IMAGE PLOTS (Range Doppler profile):

dopp = (1:size(quantimage,1))*prf/windowSize; % Doppler axis
rang = 1:size(quantimage,2);                  % Range axis
colordef black                                % Background color
figure;
Ncontours = 120;                              % # of plot contour lines
contour(rang,dopp,quantimage,Ncontours);      % Contour Plot
%title('Range-Doppler profile of a false target');
xlabel('Down Range Cells');
ylabel('Doppler frequency [Hertz]');
axis tight
figure;
colormap(hot(100))
surf(rang,dopp,quantimage), shading interp, lighting phong, view(0,90);
%title('Range-Doppler profile of a false target');
xlabel('Down Range Cells');
ylabel('Doppler frequency [Hertz]');
axis tight
```

C. CD-ROM DISTRIBUTION AND ONLINE AVAILABILITY

Due to the large size of the X3D 3D ISAR Simulation and NURBS implementation, their associated code is available only in a cd-rom. All software associated with this thesis, including the MATLAB modifications, X3D Simulation, and NURBS code is contained in the cd-rom, available upon request from the

Center for Joint Services Electronic Warfare
C/O Professor P.E. Pace
Naval Postgraduate School
Code EC/PC
Monterey, CA 93943
(831) 656-3286
pepace@nps.navy.mil.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Le Dantec, Fernando A., *Performance Analysis of a Digital Image Synthesizer as a Counter-Measure Against Inverse Synthetic Aperture Radar,* Masters Thesis, Naval Postgraduate School Monterey, 2002.

[2] Wehner, D. R., *High-Resolution Radar*, Artech House, Norwood MA, 1995.

[3] Stimson, G. W., *Introduction To Airborne Radar*, Second Edition, SciTech Publishing, Mendham NJ, 1998.

[4] Carrara, W., *Spotlight Synthetic Aperture Radar,* Artech House, Norwood MA, 1995.

[5] *Jane's Radar and Electronic Warfare Systems*, Jane's Information Group, Surrey, United Kingdom, 2001-2003.

[6] Pace, P. E., Fouts D. J., Karow C., Ekestrom S. R., "A Digital False Target Synthesizer for Countering ISAR," *IEE Proceedings – Radar, Sonar and Navigation*, pp. 248-257. October 2002.

[7] Fouts, D. J., Pace P. E., Karow C., Ekestrom S. R., "A Single-Chip False Target Radar Image Generator for Countering Wideband Image Radars," *IEEE Journal of Solid State Circuits*, pp. 751-759. June 2002.

[8] Xu, Xiaojian, Narayanan, Ram M., "Three-Dimensional Interferometric ISAR Imaging for Target Scattering Diagnosis and Modeling," *IEEE Transactions on Image Processing,* pp. 1094-1102. July 2001.

[9] Xu, Xiaojian, Luo, Hong, Huang, Peikang, "3-D Interferometric ISAR Images for Scattering Diagnosis of Complex Radar Targets," *The Record of the 1999 IEEE Radar Conference,* pp. 237-241. April 1999.

[10] Wang, Genyuan, Xia, Xiang-Gen, Chen, Victor C., "Three-Dimensional ISAR Imaging of Maneuvering Targets Using Three Receivers," *IEEE Transactions on Image Processing,* pp. 436-447. March 2001.

[11] Extensible 3D (X3D), "Part 1: Architecture and Basic Components," ISO/IEC FCD 19775-1:200x, International Draft Standard, 2003.

[12] AlphaWorks IBM Xeena Software, http://www.alphaworks.ibm.com/tech/xeena.

[13] Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE) 3D Model Library, http://www.stl.nps.navy.mil/~brutzman/Savage/contents.html.

[14] Rogers, David F., *An Introduction to NURBS with Historical Perspective,* Morgan Kaufmann Publishers, San Francisco, 2001.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. John P. Powers, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Dr. Phillip E. Pace, Code EC/Pc
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Dr. Don Brutzman
   MOVES Institute
   Naval Postgraduate School
   Monterey, California.

6. ENS. Charles Adams
   United States Navy Reserve
   Harrisburg, Pennsylvania

7. Dr. Douglas J. Fouts, Code EC/Fs
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

8. Mr. Alfred A. Di Mattesa
   Naval Research Laboratory
   Washington, D.C.

9. Mr. Gregory P. Hrin
   Naval Research Laboratory
   Washington, D.C.

10. Mr. Daniel W. Bay
    Naval Research Laboratory
    Washington, D.C.

11. Dr. Frank Klemm
Naval Research Laboratory
Washington, D.C.

12. Mr. Jeff Knowles
Naval Research Laboratory
Washington, D.C.

13. Dr. Joseph Lawrence
Office of Naval Research
Arlington, Virginia

14. Mr. Jim Talley
Office of Naval Research
Arlington, Virginia

15. Mr. Greg Tavik
Office of Naval Research, Radar Division
Arlington, Virginia

16. Mr. Mike Monsma
Office of Naval Research, Radar Division
Arlington, Virginia

17. Mr. Robert Surratt
Office of Naval Research, Radar Division
Arlington, Virginia

18. Dr. Peter Craig
Office of Naval Research
Arlington, Virginia

19. Dr. Will Cronyn
Space and Naval Warfare Systems Command
San Diego, California