UNCLASSIFIED

| AD NUMBER |
| --- |
| **ADB192220** |
| LIMITATION CHANGES |
| TO:<br>**Approved for public release; distribution is unlimited.** |
| FROM:<br>**Distribution authorized to U.S. Gov't. agencies and their contractors; Critical Technology; AUG 1994. Other requests shall be referred to Director, Army Research Lab., Attn: AMSRL-SS-SG, Adelphi, MD 20783-1197.** |
| AUTHORITY |
| **ARL Form 1 dtd 9 Mar 2010** |

THIS PAGE IS UNCLASSIFIED

ARMY RESEARCH LABORATORY

# Focusing of Dispersive Targets Using Synthetic Aperture Radar

by John McCorkle and
Lam Nguyen

ARL-TR-305 (Reprinted in March 2010) August 1994

Approved for public release; distribution unlimited.

## NOTICES

### Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

**Army Research Laboratory**
Adelphi, MD 20783-1197

**ARL-TR-305 (Reprinted in March 2010)** **August 1994**

# Focusing of Dispersive Targets Using Synthetic Aperture Radar

**by John McCorkle and Lam Nguyen**

**Approved for public release; distribution unlimited.**

# 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 instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED (From - To) |
|---|---|---|
| August 1994 | Final | 1990 to 1992 |

**4. TITLE AND SUBTITLE**
Focusing of Dispersive Targets Using Synthetic Aperture Radar

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**
P612782

**6. AUTHOR(S)**
John McCorkle and Lam Nguyen

**5d. PROJECT NUMBER**
35E55B

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
Attn: AMSRL-SS-SG
2800 Powder Mill Road
Adelphi, MD 20783-1197

**8. PERFORMING ORGANIZATION REPORT NUMBER**
ARL-TR-305 (Reprinted in March 2010)

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**13. SUPPLEMENTARY NOTES**
AMS code: P612782.H9311
DA PR: AH93

**14. ABSTRACT**
The focusing of synthetic aperture radar (SAR) data presents problems to the designer. This report addresses the SAR focusing problem with special attention to focusing an area in the near field of the synthetic aperture over a decade or more of bandwidth in a manner that preserves target resonance characteristics. A method for solving this image formation problem is described, along with a computationally efficient algorithm that is applicable to real-time processing with motion compensation. Simplified program examples are given, as well as a complete program listing that executes on several single-chip array processors simultaneously. An error analysis shows quantitatively when the depth of focus is adequate to preserve long-duration target resonance ringing effects for a given target $Q$, geometry, and bandwidth.

**15. SUBJECT TERMS**
SAR, UWB, focusing radar

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | UU | 94 | Lam Nguyen |
| Unclassified | Unclassified | Unclassified | | | 19b. TELEPHONE NUMBER *(Include area code)* (301) 394-0847 |

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

# Contents

page

1. **Introduction** ..... 5
2. **Background** ..... 7
   - *2.1 Terminology* ..... 7
   - *2.2 Approaches to Focusing* ..... 7
   - *2.3 Null Steering, Pattern Forming, and Grating Lobes* ..... 10
   - *2.4 Target Resonance Effects* ..... 11
3. **Fundamental Time-Based Physical Array Approach** ..... 12
4. **Short Impulse Response Approximation** ..... 13
   - *4.1 Theory* ..... 13
   - *4.2 Examples of Applying Approximation* ..... 15
     - 4.2.1 Case where $n = 0$ ..... 15
     - 4.2.2 Case where $n \neq 0$ ..... 15
5. **Error of Short Impulse Response Approximation** ..... 16
6. **Efficient Calculation** ..... 18
   - *6.1 Preprocessing* ..... 18
   - *6.2 Fast Index Calculation* ..... 19
     - 6.2.1 Approach ..... 19
     - 6.2.2 Solving for the Coefficients ..... 20
     - 6.2.3 Fast Polynomial Calculation ..... 23
     - 6.2.4 Coefficient Generator Program ..... 24
   - *6.3 Fast Focusing Algorithm* ..... 25
   - *6.4 Considerations for Fast Focusing Polynomial Order Selection* ..... 28
   - *6.5 Implementation Notes* ..... 30
7. **Beam Patterns and Sidelobe Structure** ..... 31
8. **Summary** ..... 40

**References** ..... 41
**Distribution** ..... 87

# Appendices

**Appendix A. Coefficient Generator Program** ..... 43
**Appendix B. UWB SAR Focusing Program** ..... 63

# Figures

1. Basic flight path and image area geometry ..... 7
2. Radar cross section of a sphere ..... 11

## Figures (cont'd)


page

3. Geometry for calculation of approximation error ..... 17
4. Error as a function of position in aperture ..... 17
5. Error as a function of $\theta$ with $R/L_s = 2$ ..... 17
6. Error as a function of $\theta$ with $R/L_s = 6$ ..... 17
7. Error at right aperture end as a function of $m$, where $\tau = 0$ ..... 17
8. Error at right aperture end as a function of $m$, where $\tau = 4\alpha$ ..... 17
9. Image partitioning and notation ..... 24
10. Coefficient generator flow chart ..... 24
11. 50-MHz point reflector, equal weighting, end view ..... 32
12. 50-MHz point reflector, Hamming weighting, end view ..... 32
13. 50-MHz point reflector, equal weighting, above axis ..... 33
14. 50-MHz point reflector, Hamming weighting, above axis ..... 33
15. 200-MHz point reflector, equal weighting, end view ..... 34
16. 200-MHz point reflector, Hamming weighting, end view ..... 34
17. 200-MHz point reflector, equal weighting, above axis ..... 35
18. 200-MHz point reflector, Hamming weighting, above axis ..... 35
19. 400-MHz point reflector, Hamming weighting, end view ..... 36
20. 400-MHz point reflector, Hamming weighting, end view ..... 36
21. 400-MHz point reflector, equal weighting, above axis ..... 37
22. 400-MHz point reflector, Hamming weighting, above axis ..... 37
23. 900-MHz point reflector, equal weighting, end view ..... 38
24. 900-MHz point reflector, Hamming weighting, end view ..... 38
25. 900-MHz point reflector, equal weighting, above axis ..... 39
26. 900-MHz point reflector, Hamming weighting, above axis ..... 39


## Tables


1. Analysis nomenclature ..... 8
2. Computational load ..... 29
3. Matrix showing computational load versus partitioning for $j' = 3$ and $J = 2304$ ..... 29


## Displays


1. Subroutine Inner_Loop ..... 25
2. Subroutine Middle_Loop ..... 26
3. Subroutine Outer_Loop ..... 27

# 1. Introduction

Over the past 40 years, studies have been made [1–4] of the process of obtaining images of the reflectivity or density of target areas that are rotating and translating with respect to a sensor such as a monostatic or bistatic radar, a sonar, or an x-ray CAT scanner. Ausherman et al [5] have written an excellent review of the work done in this area. Target areas that rotate and translate relative to a radar include, for example, planet surfaces observed from a satellite, ground terrain observed from airborne platforms, subsurface objects and voids observed from moving vehicles, and people scanned by a rotating x-ray system. Although the processing described is applicable to other systems, this report treats the topic from the point of view of a synthetic aperture radar (SAR).

In the SAR application, as the aspect angle between the sensor and a target changes with time, the sensor collects a sequence of signal records. After data are collected for $T_s$ seconds, the aspect angle has changed by $\theta_s$ degrees. These received signals are then coherently processed to produce the reflectivity profile of the target area. Down-range resolution into range bins is determined primarily by the bandwidth of the sensor. Cross-range resolution is obtained primarily by coherent processing of the received signals so that a very wide aperture is simulated: an aperture that is $\theta_s$ degrees wide.

Implementation and study of image formation processing have been limited in two ways. First, the image formation processing has assumed that targets are isotropic point scatterers, although many targets are anisotropic resonant scatterers. Treatment of resonant scattering becomes important when the sensor spectrum covers the Rayleigh, resonant, and optical regions of a family of targets. For example, discrimination between scatterers can be based on the unique signature of each target. But in order for the discrimination to work in the context of microwave reflectivity imaging, the information in the signature must be preserved during the image formation process.

The second limitation in previous work is that systems with relatively narrow bandwidth have been assumed. For example, the compressed pulse width of the sensor is assumed to be at least several and usually many cycles of an rf carrier frequency. So a single range bin is derived from several cycles of the carrier. Newer UWB (ultra-wide-bandwidth) sensors, however, have made it possible to make the image range-bin size roughly half the wavelength of the highest frequency in the sensor's spectrum. The bin size is much smaller (1/10 or less) than the wavelength of the lowest frequency in the spectrum. This wide bandwidth exacerbates range walk and wavefront curvature errors to the point where conventional fast Fourier transform (FFT) based processing must be restricted to very small patches within an image area.

These two limitations are not independent. It is bandwidth that allows target discrimination based on signature analysis, and it is bandwidth that makes image formation more difficult. The purpose of this report is to examine image formation processing that preserves resonant target signatures and to present an efficient method of solving the microwave reflectivity imaging problem for UWB signals and resonant targets.

# 2. Background

## 2.1 Terminology

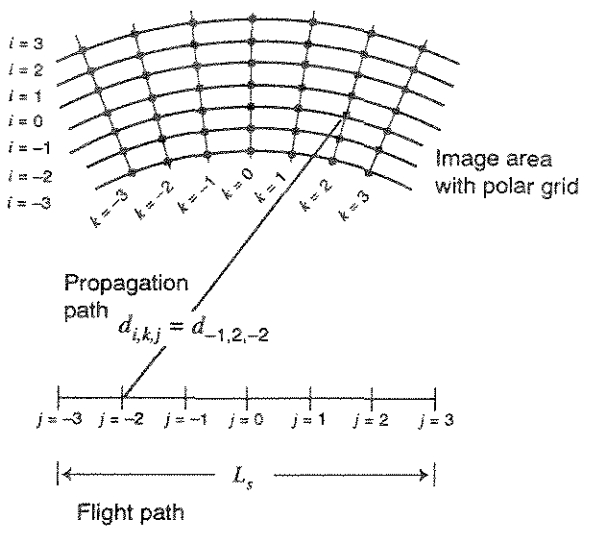
SAR systems depend upon collecting data coherently along a path. This path is referred to as the "synthetic aperture." Figure 1 shows a sketch of the scenario projected onto a plane. An aircraft carrying the SAR system flies at some elevation $h$ above the ground, over a distance $L_s$, to form the synthetic aperture. The image area grid is referenced to the center of the aperture. Range is marked off by the parameter $i$. The parameter $k$ marks off azimuthal (bearing) lines. The sample points in the aperture are marked off by the parameter $j$. The distance between the $j^{th}$ point in the aperture and the $(i,k)^{th}$ position in the image area is denoted by $d_{i,j,k}$. Both the azimuthal lines and the range lines are referenced to the center of the aperture.* Table 1 summarizes the nomenclature used in this report.

(Although nearly all operational SAR systems use a rectangular grid, the advantages of a polar grid are that for a ringing target, the grid provides equally spaced samples of the ring-down, and these samples are optimally focused and on a single bearing line. These features make it easy for postprocessors to look for specific ringing signatures.)

## 2.2 Approaches to Focusing

A "Doppler" paradigm is often used in discussions of SAR processing, or SAR focusing. Referring to figure 1, assume that the radar platform is moving at velocity $v$ as it collects data along the aperture. Therefore, the data collection points are spaced equally in time, occurring at a rate governed by $v$ and the PRF. (This sampling across the aperture is sometimes referred to

**Figure 1. Basic flight path and image area geometry.**




* *This grid is convenient for postprocessing. An x-y grid can also be computed with the techniques described.*

**Table 1. Analysis nomenclature.**

| Symbol | Meaning |
|---|---|
| $L_s$ | Length (meters) of the synthetic aperture. |
| $s_j(t)$ | Denotes a received signal amplitude (volts) as a function of time (seconds from the leading edge of the transmitted pulse), at the $j^{th}$ position in the aperture. The analog-to-digital outputs represent this signal. |
| Caret—e.g., $\hat{s}(t)$ | Denotes that the function is an estimate. The example denotes an estimate for a received signal. |
| Subscript $i,k,j$ | Particularizes the function to be at particular geometries, like the $j^{th}$ position in the synthetic aperture, or the $k^{th}$ bearing line, or the $(i,k)^{th}$ position in the image area. |
| $d_{i,j,k}$ | Denotes the distance (meters) between the $j^{th}$ position of the radar and the $(i,k)^{th}$ position in the area to be imaged. |
| $t_{i,k,j} = \frac{2d_{i,k,j}}{c}$ | Denotes the round-trip time (seconds) for the radio energy to travel from the $j^{th}$ position in the synthetic aperture, to the $(i,k)^{th}$ position in the image area, and back. |
| $c$ | The speed of light. |
| $\alpha = t_{i+1,k,j} - t_{i,k,j}$ | Time shift between range bin $i$ and range bin $i + 1$ at the center of the aperture (where $j = 0$). |
| PRF | Pulse repetition frequency of the radar in hertz. |
| $\mu$ | Relative bandwidth $\mu = (F_{hi} - F_{lo})/F_0$ and $F_0 = (F_{hi} + F_{lo})/2$. |
| UWB | Ultra-wide bandwidth, where $\mu \approx 1$. |
| $\lambda$ | Wavelength in meters. |
| $f_{i,k}(t)$ | Impulse response of target at $(i,k)$. |

as "slow time.") If the radar is operating at a frequency $f_0$, then the echo from a target in the image area will have a Doppler shift profile as the platform moves through the aperture. The shift will be upward while the platform approaches, it will drop to zero as the platform moves to a position broadside to the target, and it will be downward as the platform recedes. Every target position will have a unique Doppler profile. Since every position in the image has a distinct Doppler profile, one can form an image by simply assigning to each pixel a filter matched to the Doppler profile expected for a target at the location represented by that pixel.

The classic "polar formatting" [6] approach to computing a SAR image adjusts (time shifts) the data at each aperture point so that the new data set appears as if the platform had moved in a short circular path, with the circle at the center of the image area. Once this formatting is done, a target at the center point has the unique Doppler profile of zero over the entire aperture. Other points have other unique Doppler profiles.

A Fourier transform is typically used to recover the image from the Doppler profiles. This approach works well as long as the circular path is sufficiently short, the image area sufficiently small, the signal bandwidth sufficiently small, and the distance from the radar to the image center sufficiently long. This report addresses the case where none of these restrictions apply.

Although the Doppler paradigm has helped countless people to visualize SAR focusing, it is not necessary, nor always helpful, in solving the SAR focusing problem. For example, Doppler shift, which is defined as $2v/\lambda$, works fine when $\lambda$ varies by a few percentage points, but it becomes a stumbling block in the UWB case where $\lambda$ can vary by 10 or 100 to 1. Thus, although Doppler has proven to be a very convenient narrow-band concept, its convenience breaks down at wide relative bandwidths.

The focusing problem can also, however, be looked at as a stationary array of $N$ antennas whose outputs are digitally stored and combined in a computer to form beams. We believe that the focusing problem is easier to visualize and solve within this stationary array paradigm when neither the geometry nor the bandwidth are restricted. In this paradigm one can see that equation (1) coherently focuses the SAR data by summing across the array:

$$f_{i,k}(t) = \sum_j s_j(T_{i,k,j} + t) \text{ for } t \geq 0 \quad . \tag{1}$$

Both the Doppler and stationary array approaches can be seen to be identical at the center point of the "polar format" patch. Notice that regardless of carrier frequency, the center point of the polar formatted data is always analyzed by the dc term of the Fourier transform, which is simply a summation of the data points. The summation shown in equation (1) is identical to finding the dc term of the polar formatted SAR data. But instead of formatting once and then finding many "Doppler" profiles via an FFT, equation (1) formats and "sums" the data many times; each formatting makes a different pixel the center, and then the dc term summation is used to calculate the value for that pixel. The result is that optimally focused beams are formed—beams taking full advantage of both the entire aperture and the entire signal bandwidth. The focusing is truly frequency independent.

How does the beam width compare with that obtained from "conventional" (far-field narrow-band) antenna theory? The rule-of-thumb half-power beamwidth of a line array is approximately $\lambda/L$, where $L$ is the length of the array. The beams formed by the processing described above follow this rule and have a width proportional to $\lambda$. Low frequencies have wide beams, and high frequencies have narrow beams. This fact has interesting consequences in the time domain as a source moves through a beam. Although the impulse response at the center of the beam is a narrow pulse, as one moves away from the center, the impulse response broadens. This time-domain broadening occurs because more and more high-frequency energy is lost as the source moves out of the narrowing high-frequency beam.

The switch from the Doppler to the stationary array approach may lead to thinking in "phased array" terms. Such an approach is a mistake when bandwidth and/or geometry are not restricted. Whenever the geometry is not restricted to the far field of an aperture, plane-wave simplifications

break down. This breakdown invalidates simple phase steering. Since a Fourier transform forms beams by simple phase steering, Fourier techniques becomes less and less useful as targets move into the near field. In a UWB system, phase becomes meaningless with regard to defining element positions or the beam-forming network. To speak of shifting one element 180° with respect to another element implies a fixed $\lambda$. If, for example, $\lambda$ changed by 2 to 1, then a delay line that provided 180° at one frequency would provide 360° at the other. Yet delay lines are precisely the element needed to build a wide-bandwidth "phased-array" antenna. When $\lambda$ varies several octaves, the best parameters to use in the equations defining the antenna are the time and distance: the time shift of the delay lines that form the beam-forming network, and distance between antenna elements. Thus it is best to think in "timed-array" terms. This time-based framework results in derivations that are frequency and geometry independent.

## 2.3 Null Steering, Pattern Forming, and Grating Lobes

Generally, an antenna designer would say that the most critical aspect of making desirable antenna patterns is forming nulls. A simple classic case is spacing two elements at 90° and phasing them by 90° to form an endfire beam in one direction and a null in the opposite direction. In the simplest case (where the element weighting is +1 and –1), to form a UWB null one could time-steer the array to where the null should be, and then invert half the elements before summing. Of course, using other weighting factors allows more freedom. An impulse signal $\delta(t)$ coming from a direction other than the null would produce in the receiver some array-induced waveform. If we ignore bandwidth effects from each element, that waveform would be a function of the + and – weighting and element spacing. By definition, that waveform is the antenna-array impulse response for that beam angle. Matched filtering to that waveform forms a main lobe in that direction.

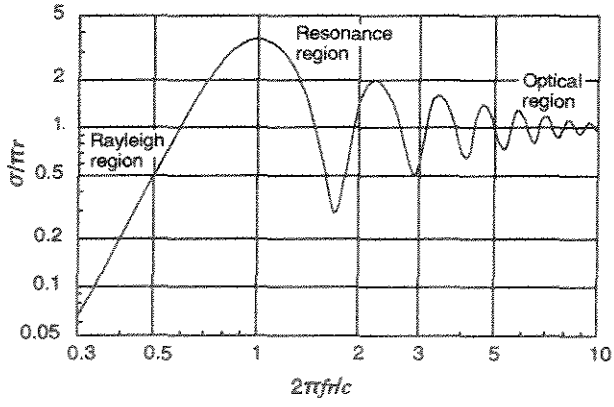
Grating lobes are customarily defined as lobes whose gain is equal to the main beam. However, this definition leads to confusion in the UWB case. Are there grating lobes? That depends. If the response to the UWB signal is seen as an impulse, then the answer is no. Since the peak response on the main lobe is higher than the response at any other angle, the definition fails. On the other hand, the antenna is a linear system. It behaves just like an identical array operating at a single frequency. So, if the response is to a cw signal, the answer can be yes. If the elements are physically spaced more than $\lambda/2$ apart at the highest frequency component in the UWB waveform, then there certainly will be a grating lobe at that frequency component. All the insight gained from cw antenna analysis holds and remains useful. One must, however, be careful when applying terms like *grating lobes* that presuppose a narrow-band signal. For SAR, the element spacing (velocity/PRF) needed will be a function of what sidelobes are permissible at the various frequency components in the UWB waveform.

## 2.4 Target Resonance Effects

Historically, the relative bandwidth of radars has been sufficiently small that a target's echo is adequately modeled by a single number, $\sigma$, the radar cross section (RCS), usually given in square meters. When $\mu \geq 0.5$, however, a single number may no longer be adequate—$\sigma$ is a function of frequency. For example, figure 2 is a plot of the RCS of a sphere. In addition to the magnitude characteristic plotted, there is also a phase characteristic. These two frequency-domain characteristics can also be represented in the time domain by a ringing or resonant response. In either case—time domain or frequency domain—the waveform in the plots can be referred to as the impulse response of the target.

Since historical SAR systems have not been designed to respond to target ringing, no attention was paid to preserving the ringing information. This report describes and analyzes a procedure to focus a large array, over ultra-wide bandwidths, in such a way as to preserve the resonant response of targets, even when they are in the near field.

**Figure 2. Radar cross section of a sphere.**



# 3. Fundamental Time-Based Physical Array Approach

Consider an aperture looking at completely empty space except for an isotropic scatterer at position $(i,k)$. Also assume that an ideal impulse $\delta(t)$ is broadcast. It is desirable to take advantage of all the echo energy across the aperture. To do that, one must sum the energy from all collection points along the synthetic aperture. However, this summation must be done so that the energy adds coherently at all frequencies. Frequency-independent adding is accomplished as

$$f_{i,k}(t) = \sum_j s_j(T_{i,k,j} + t)\, w_j \text{ for } t \geq 0 \quad . \tag{1}$$

Here, $f_{i,k}(t)$ would be the impulse response of the target located at position $(i,k)$. Note that the $T_{i,k,j}$ term time shifts the received signals $s_j$ so that the target impulse response starts at $t = 0$ at all points in the aperture. The $w_j$ term is simply an amplitude tapper across the aperture.

The above discussion assumes that a target's impulse response is sufficiently similar at all points along the aperture for equation (1) to be considered true. The response of a vertical dipole, for example, does not change depending on where the radar is positioned in the aperture. It is therefore an isotropic target.

Suppose we relax the isotropic requirement, and suppose that a complex scatterer is at position $(i,k)$ in the image area. A horizontal dipole, for example, is an anisotropic target. We could take advantage of the anisotropic behavior to extend the detection and target recognition performance of the radar by rewriting equation (1) as

$$f_{i,k}(t) = \sum_j y_j(T_{i,k,j} + t) \text{ for } t \geq 0 \quad , \tag{2}$$

where $y_j(t) = s_j(t) \otimes x_j(t)$ and the convolution step represents filtering. In this case, a set of filters $X_j(\omega)$ would be needed, each one matched to the target response at the bearing of the $j^{th}$ position in the aperture. In order to simplify the rest of this report, we use equation (1) as the fundamental equation. Nonetheless, one must recognize that real targets are complex anisotropic polarimetric scatterers; this fact should not be ignored for large arrays.

# 4. Short Impulse Response Approximation

A typical 2-D SAR image is simply the echo magnitude mapped to intensity. Equation (1) expands the typical 2-D image into a 3-D image with the target ringing along the third dimension. Given the heavy computational load of typical SAR processing, if it is required that an $f(t)$ (rather than a single value) be computed for every pixel, then a massive computer would be needed. The impact of this computational load is even worse when one considers that the mass of resulting data must be analyzed in the target-detection/identification phase. This section defines an approximation that reduces the problem back to a 2-D case and presents an error analysis of the approximation.

## 4.1 Theory

The problem is that we have added a third dimension to measure target ringing. Note, however, that if there was only one aperture position, then ringing of the target would appear in pixels only behind the target. Even with an aperture of many positions, ringing will appear behind the target. But since the geometry is not constrained, and near-field operation is presumed, the antenna beam defocuses behind the target.

Equation (1) will perfectly focus ringing of unbounded duration. For practical purposes, however, the target rings only for a finite duration, say $M$ range bins. The question is, given that the ringing is finite in duration, can the ring information can be retained in a 2-D image? In other words, suppose that instead of calculating a time series $f_{i,k}(t)$ for each pixel in the image, only one value is calculated, for example, $f_{i,k}(\tau)$, where $\tau$ is fixed for the image. If the ringing can be retained in this 2-D image, then computational load is greatly reduced. The aim of this section is to describe and quantify the error bounds when a 2-D image $f_{i,k}(\tau)$ is used.

First, define $\alpha$ to be the round-trip time it takes the radar pulse to traverse one range bin on the polar grid; that is,

$$\alpha = t_{i+1,k,j} - t_{i,k,j} : \forall k, \text{ and } j = 0 \quad . \tag{3}$$

Since the grid for the image has been defined to be referenced to the center of the aperture (the $j = 0$ position), the $i$ parameter can be thought of as a quantized time $t$ parameter. While $t$ is in units of seconds, $i$ is in units of range bins, and they are related by $\alpha$—one range bin equals $\alpha$ seconds. Now we can write

$$s_j(T_{i,k,j} + m\alpha) \equiv s_j(T_{i+m,k,j}) \text{ for } \begin{Bmatrix} \text{case 1: } j = 0, \forall m, \forall i, \forall k \\ \text{case 2: } m = 0, \forall j, \forall i, \forall k \end{Bmatrix} \quad . \tag{4}$$

We make an approximation to equation (4), generalizing it to include all aperture points over a limited range of $m$ to arrive at

$$s_j(T_{i,k,j} + m\alpha) \approx s_j(T_{i+m,k,j}) \text{ for } \forall i, \forall k, \forall j, m = -M \ldots M \quad . \tag{5}$$

The following may be said of the approximation in equation (5).

1. It is perfect at the center of the aperture ($j = 0$) regardless of $m$.
2. It is perfect at $m = 0$ regardless of $j$.
3. It gets worse as $m$ deviates further from zero.
4. It is worst at the end points of the aperture.

A solution is desired to equation (1) in the form of $f_{i,k}(\tau)$, where $\tau$ is a constant. In a practical system, $\tau$ will be mapped to discrete range bins, so let

$$\tau = n\alpha \quad . \tag{6}$$

Substituting equations (5) and (6) into (1), we define a 2-D "image" $f(i,k)$ as

$$\begin{aligned} f_{i,k}(0) &= \sum_j s_j(T_{i,k,j}) \\ &\approx f_{i-n,k}(n\alpha) = \sum_j s_j(T_{i-n,k,j} + n\alpha) \equiv f(i,k) \quad . \end{aligned} \tag{7}$$

If one thinks of a ringing target, then equation (7) can be described as allowing the point of perfect focus to be adjusted to any depth $n$ in the ring. For example, if $n = 0$, then the perfect focus point would be at the leading edge (the first sample) of the ringing response. If $n \neq 0$, say $n = 3$, then the third sample of the ringing response would be perfectly focused.

Next consider the indexing. The indexing is performed such that $f(i,k)$ always represents the leading edge of the response from a target located at $(i,k)$ regardless of whether it is perfectly focused or not. Once $f(i,k)$ is found, we now wish to find the discrete samples of the target ringing. The samples will be counted as $m = 0$ for the first sample, $m = 1$ for the second, and so on. We obtain these samples by simply incrementing the $i$ index. The $m^{\text{th}}$ value is just $f(i + m,k)$, which follows from equations (5) and (7) as

$$\begin{aligned} f_{i,k}(m\alpha) &\approx f_{i-n+m,k}(n\alpha) = f(i+m,k) \quad , \text{ or} \\ \sum_j s_j(T_{i,k,j} + m\alpha) &\approx f_{i-n+m,k}(n\alpha) = f(i+m,k) \quad . \end{aligned} \tag{8}$$

Clearly equation (8) is identical to equation (1) (i.e., perfect) when $m = n$. So equation (8) gives perfect focus at $m = n$. The name "short impulse response approximation" is used because the approximation needs to remain accurate only over the duration of a target's impulse response.

## 4.2 Examples of Applying Approximation

To illustrate the use of equation (8), we follow these steps:

1. Fix $n$.
2. Place a target (for the purposes of the illustration) at say $i = 237$ in range on the $k^{\text{th}}$ bearing.
3. Use equation (7) to calculate $f(i,k)$ for all $(i,k)$.

The cases of interest are where $n = 0$ and where $n \neq 0$. Let us consider each separately.

### *4.2.1 Case where n = 0*

Where $n = 0$, use equation (8) to find the ringing response of the target. The response for the first three points of the focused impulse response is

$$
\begin{aligned}
f_{237,k}(0) &\approx f_{237-n+m,k}(0) = f_{235,k}(0) = f(237,k) \quad \text{(case for } m = 0) \ , \\
f_{237,k}(\alpha) &\approx f_{237-n+m,k}(0) = f_{236,k}(0) = f(238,k) \quad \text{(case for } m = 1) \ , \\
f_{237,k}(2\alpha) &= f_{237-n+m,k}(0) = f_{237,k}(0) = f(239,k) \quad \text{(case for } m = 2) \ .
\end{aligned}
$$

Since $n = 0$, the amplitude of the leading edge ($m = 0$) of the impulse response of that target is perfectly focused. As $m$ increases, the approximation gets worse. So the approximation is useful as long as the target resonance dies before the approximation gets too bad.

### *4.2.2 Case where n ≠ 0*

Where $n \neq 0$, perfect focus is $n\alpha$ seconds past the leading edge of the target impulse response. Suppose, for this example, $n = 2$. The first four data points for the impulse response are

$$
\begin{aligned}
f_{237,k}(0) &\approx f_{237-n+m,k}(n\alpha) = f_{235,k}(2\alpha) = f(237,k) \quad \text{(case for } m = 0) \ , \\
f_{237,k}(\alpha) &\approx f_{237-n+m,k}(n\alpha) = f_{236,k}(2\alpha) = f(238,k) \quad \text{(case for } m = 1) \ , \\
f_{237,k}(2\alpha) &= f_{237-n+m,k}(n\alpha) = f_{237,k}(2\alpha) = f(239,k) \quad \text{(case for } m = 2) \ , \\
f_{237,k}(3\alpha) &\approx f_{237-n+m,k}(n\alpha) = f_{238,k}(2\alpha) = f(240,k) \quad \text{(case for } m = 3) \ .
\end{aligned}
$$

Note that the leading edge is not perfectly focused, as it was when $n = 0$. The important point here is that one can choose $n \neq 0$ to allow the leading edge to defocus slightly for the sake of keeping later points in better focus.

# 5. Error of Short Impulse Response Approximation

Landt, Miller, and Van Blaricum [7] show the transient echo response of a thin (high-$Q$) dipole. Its ringing is damped after five cycles. This response allows one to gain an intuitive idea of the kind of range needed in the approximation. A 1/2-ft dipole should ring for five cycles at 1 GHz. If the A/D sampler collects data at 2 Gs/s and $\tau = 0$, then at $m = 10$, data for all five cycles will have been collected. If the dipole were 5 ft long, then it would ring for five cycles at 100 MHz. So at $m = 100$, data for the five cycles will be collected. Generally, high frequencies damp quickly and low frequencies damp slowly.

Figure 3 illustrates the geometry of the approximation. The discussion assumes this geometry. If a target is located at $(i,k) = (237,0)$, then $R$ is the distance from the center of the array to the target; $a_1$ is the distance from the end of the array to the target; and $f_{237,0}(0)$ is the perfectly focused data point for the impulse response of the target. The next point ($m = 1$) in the impulse response is approximated by $a_2 \approx a_1 + \alpha_d$, or in general $a_2 \approx a_1 + m\alpha_d$. If $\varepsilon$ is the difference (error) between the approximation and the actual, then

$$\begin{aligned}\varepsilon &= \left(a_1 + m\alpha_d\right) - a_2 \\ &= \sqrt{R^2 + \frac{L_s}{2} - L_s R\cos\theta} + m\alpha_d - \sqrt{(R + m\alpha_d)^2 + \frac{L_s}{2} - L_s\left(R + m\alpha_d\right)\cos\theta} \quad . \end{aligned} \tag{9}$$

As an example of finding the approximation error, suppose $R = 2$ km, $\theta = 90°$, $L_s = 1$ km, and $m = 10$. How many degrees off (round trip) would the end points be at 1 GHz? Plugging these numbers into equation (9), we find $\varepsilon = 0.0447$ m. So the round trip phase error at 1 GHz is 107°. Figure 4 is a plot of the error as a function of the position in the aperture for $\theta = 90°$, 76°, 50°, and 30°. A peculiar characteristic is that the error peaks at 76°. This peaking is a result of working at a range of only twice the aperture length. Figures 5 and 6 show the error as a function of the beam angle $\theta$, for $m = 5$, 10, 15, and 20 at two ranges.

Figure 7 is a plot of the error at the right end point of the aperture as a function of $m$, with $\tau = 0$. Since the error at the right end is greater than the error at the left end for $\theta$ angles below 90°, this is the worst case. Figure 8 is the same plot but with $\tau = n\alpha$ and $n = 4$. In this case, the leading edge of the impulse response (at $m = 0$) is out of focus by about 45° at 1 GHz. Perfect focus occurs when $m = n = 4$.

Figures 4 through 8 give an indication of the bounds over which a five-cycle ring of a high-$Q$ scatterer is adequately captured by the approximation used in equation (4). These figures also show that making $\tau \neq 0$ can double the depth of focus on resonant targets. The optimum $\tau$, however, depends on the frequency and $Q$ of the resonance.

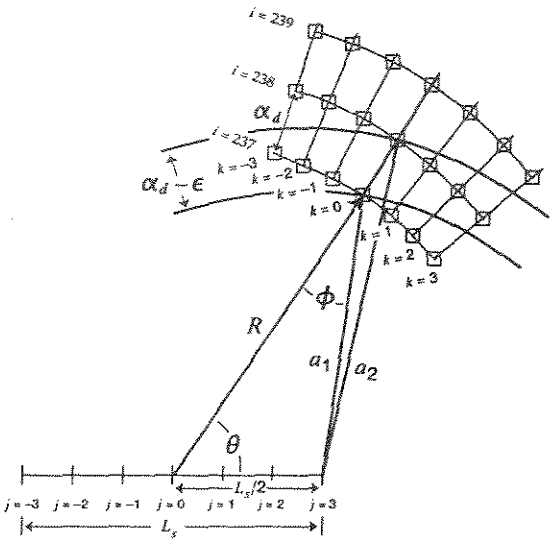



**Figure 3. Geometry for calculation of approximation error.**

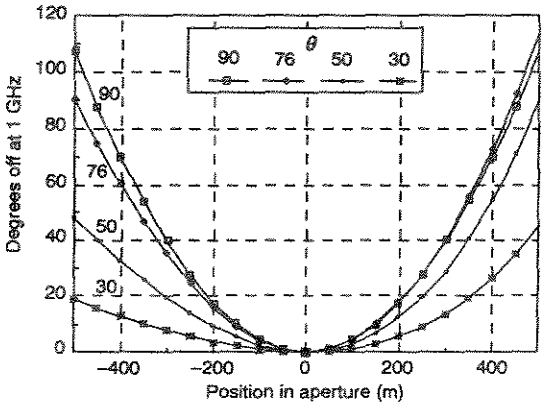



**Figure 4. Error as a function of position in aperture.**

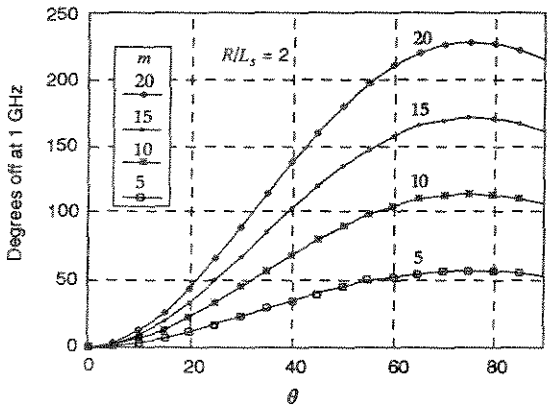



**Figure 5. Error as a function of $\theta$ with $R/L_s = 2$.**

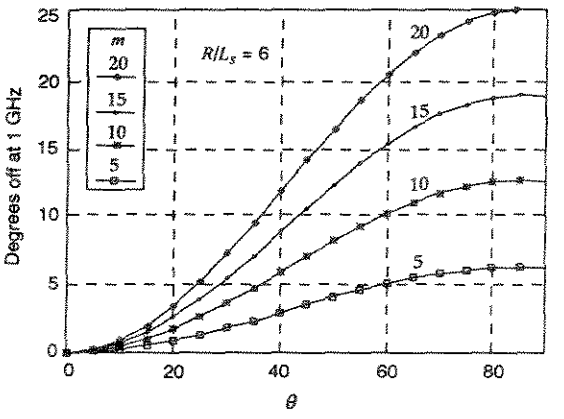



**Figure 6. Error as a function of $\theta$ with $R/L_s = 6$.**

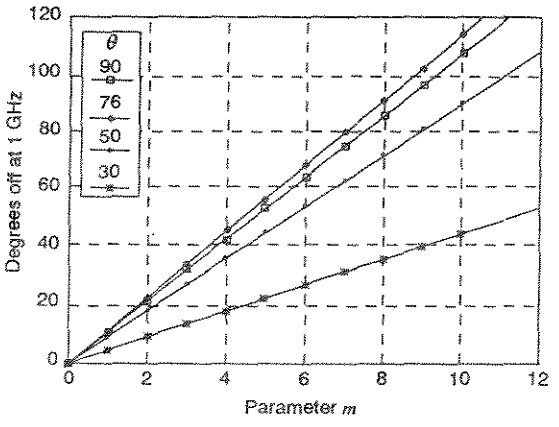



**Figure 7. Error at right aperture end as a function of $m$, where $\tau = 0$.**

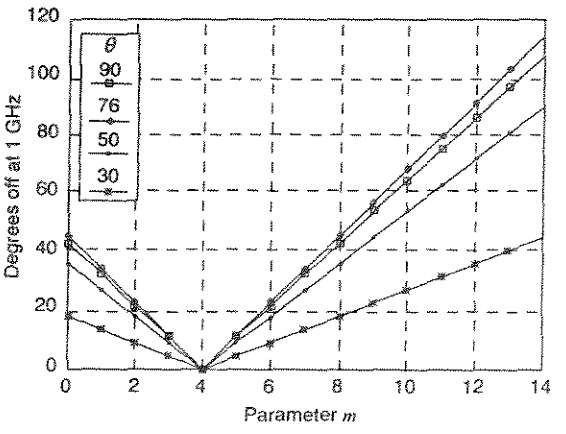



**Figure 8. Error at right aperture end as a function of $m$ where $\tau = 4\alpha$.**

# 6. Efficient Calculation

Fast focusing can be broken into three stages: (1) preprocessing raw data, where interpolation is performed; (2) computing a set of polynomial coefficients that will be used for fast index calculation; and (3) performing the focusing summation of equation (7) using the coefficients found in stage 2 to find the index corresponding to the proper time shift.

## 6.1 Preprocessing

A method to efficiently implement time shifting is needed. The SAR data are collected by an A/D converter that outputs a vector of $N$ numbers (voltages) at each aperture position. We call this vector $\tilde{s}_j(\tilde{i})$. A time shift is, therefore, simply a shift in the index $\tilde{i}$.

Typically, the time shifting obtained by indexing on the original $N$-point vector is not fine enough. Finer resolution is gained by a two-stage interpolator. First, a high-quality interpolator is used to produce a new $K$-point vector $s_j(i)$. It is usually implemented by insertion of $Z - 1$ zeros between each data point in the original vector, after which the new sequence is passed through a low-pass FIR (finite impulse response) filter. The process results in a new vector with nearly the desired time-shift resolution. The length, in this case, is $K = ZN$. We can gain extra fine resolution by using a floating-point index with simple linear interpolation to find a value between any two data points in the interpolated data vector. For example, if the index $i$ were 6.3, then the interpolated value would just be $0.7s_j(6) + 0.3s_j(7)$.

The focusing algorithm that follows uses these techniques in the following sequence, which is typically referred to as back projection.

1. Read in one $N$-point vector $\tilde{s}_j(\tilde{i})$ of raw data.
2. Do an $Z$-point interpolation, to form a new $K$-point vector $s_j(i)$.
3. Iterate for all pixels:
   a. Find the floating point index needed for a pixel.
   b. Find the data value for that pixel either by rounding the index and grabbing the value, or by linearly interpolating between adjacent values.
   c. Sum into that pixel the data value obtained.

## 6.2 Fast Index Calculation

### 6.2.1 *Approach*

Typically, the greatest computational load in back-projection focusing is in finding the index. Let $P(i,k,j)$ be the exact index needed. Then equation (7), the 2-D focusing equation, becomes

$$f_{i,k} = \sum_j s_j(P(i,k,j)) \quad . \tag{10}$$

Calculating $P(i,k,j)$ exactly involves 3-D trigonometric solutions for every pixel in the image at every position in the aperture. Such a calculation would be prohibitively large. For practical implementation, a secondary approximation is used to speed the computations. Extremely efficient computational methods exist for finding evenly spaced solutions to polynomials. Therefore, our approach is to define a polynomial in three variables $G(i,k,j)$, where $G(i,k,j) \approx P(i,k,j)$, and use $G$ to compute the index.

The first issue that arises is choosing the order of the polynomial needed for each variable. To understand the method, suppose that a second-degree polynomial is adequate for each of the three variables $i,k,j$. Now the problem can be restated as follows: for a given image pixel $(i,k)$, and a given aperture position $(j)$, find the coefficients $a_m$ for $m = 0...26$ such that

$$\begin{aligned}
G(i,k,j) &\approx P(i,k,j) \\
&= q_{0,k,j} + q_{1,k,j}i + q_{2,k,j}i^2 \\
&= [c_{0,j} + c_{1,j}k + c_{2,j}k^2] + [c_{3,j} + c_{4,j}k + c_{5,j}k^2]i + [c_{6,j} + c_{7,j}k + c_{8,j}k^2]i^2 \\
&= [(a_0 + a_1 j + a_2 j^2) + (a_3 + a_4 j + a_5 j^2)k + (a_6 + a_7 j + a_8 j^2)k^2] \\
&+ [(a_9 + a_{10} j + a_{11} j^2) + (a_{12} + a_{13} j + a_{14} j^2)k + (a_{15} + a_{16} j + a_{17} j^2)k^2]i \\
&+ [(a_{18} + a_{19} j + a_{20} j^2) + (a_{21} + a_{22} j + a_{23} j^2)k + (a_{24} + a_{25} j + a_{26} j^2)k^2]i^2 \quad .
\end{aligned} \tag{11}$$

With the equation written in this format, it is easy to see that one can code the index calculation as loops nested three deep, with $j$ as the outer loop, $k$ as the middle loop, and $i$ as the inner loop.

### 6.2.2 *Solving for the Coefficients*

The coefficients $a_m$ can be found numerically. The approach is to calculate the exact index required for $M$ points in the image at each of $H$ positions in the aperture, and do a least-squares fit to find the coefficients $a_m$ for $m$ = 0...26. We can find a generic solution to this problem, for arbitrary image size, by using a pseudo-inverse operation to perform the least-squares fit. When matrix $\mathbf{A}$ is not square but rectangular, then $\mathbf{A}^{-1}$ is known as the pseudo inverse and is defined as

$$\mathbf{A}^{-1} = (\mathbf{A}^T \bullet \mathbf{A})^{-1} \bullet \mathbf{A}^T . \tag{12}$$

An example explains the method. To simplify the example, we find a solution for only a single position in the aperture at $j = 0$. So we find the $c_0$ through $c_8$ needed to calculate the index needed for $s_0(G(i,k,0))$. Take $M$ = 25 points on a patch to be focused: five ranges (close in to far out: $i = 0, b, 2b, 3b, 4b$) on each of five azimuths (left side to right side: $k = 0, a, 2a, 3a, 4a$). The vector $\vec{\mathbf{B}}$ is the exact (floating-point) calculated index needed for those 25 points in the image. The matrix $\mathbf{A}$ is set up so that the width of the patch is $4a$ and the depth of the patch is $4b$. So we have

$$G(i,k,j)\big|_{j=0} = c_{0,0} + c_{1,0}k + c_{2,0}k^2 + c_{3,0}i + c_{4,0}ik + c_{5,0}ik^2 + c_{6,0}i^2 + c_{7,0}i^2k + c_{8,0}i^2k^2 \tag{13}$$

or

$$G(i,k,j) = \begin{bmatrix} 1 & k & k^2 & i & ik & ik^2 & i^2 & i^2k & i^2k^2 \end{bmatrix} \bullet \vec{\mathbf{C}}_j$$

with

$$\vec{\mathbf{A}} \bullet \vec{\mathbf{C}} = \vec{\mathbf{B}} , \tag{14}$$

which is

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & a & a^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2a & 4a^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 3a & 9a^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 4a & 16a^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & b & 0 & 0 & b^2 & 0 & 0 \\
1 & a & a^2 & b & ab & a^2b & b^2 & ab^2 & a^2b^2 \\
1 & 2a & 4a^2 & b & 2ab & 4a^2b & b^2 & 2ab^2 & 4a^2b^2 \\
1 & 3a & 9a^2 & b & 3ab & 9a^2b & b^2 & 3ab^2 & 9a^2b^2 \\
1 & 4a & 16a^2 & b & 4ab & 16a^2b & b^2 & 4ab^2 & 16a^2b^2 \\
1 & 0 & 0 & 2b & 0 & 0 & 4b^2 & 0 & 0 \\
1 & a & a^2 & 2b & 2ab & 2a^2b & 4b^2 & 4ab^2 & 4a^2b^2 \\
1 & 2a & 4a^2 & 2b & 4ab & 8a^2b & 4b^2 & 8ab^2 & 16a^2b^2 \\
1 & 3a & 9a^2 & 2b & 6ab & 18a^2b & 4b^2 & 12ab^2 & 36a^2b^2 \\
1 & 4a & 16a^2 & 2b & 8ab & 32a^2b & 4b^2 & 16ab^2 & 64a^2b^2 \\
1 & 0 & 0 & 3b & 0 & 0 & 9b^2 & 0 & 0 \\
1 & a & a^2 & 3b & 3ab & 3a^2b & 9b^2 & 9ab^2 & 9a^2b^2 \\
1 & 2a & 4a^2 & 3b & 6ab & 12a^2b & 9b^2 & 18ab^2 & 36a^2b^2 \\
1 & 3a & 9a^2 & 3b & 9ab & 27a^2b & 9b^2 & 27ab^2 & 81a^2b^2 \\
1 & 4a & 16a^2 & 3b & 12ab & 48a^2b & 9b^2 & 36ab^2 & 144a^2b^2 \\
1 & 0 & 0 & 4b & 0 & 0 & 16b^2 & 0 & 0 \\
1 & a & a^2 & 4b & 4ab & 4a^2b & 16b^2 & 16ab^2 & 16a^2b^2 \\
1 & 2a & 4a^2 & 4b & 8ab & 16a^2b & 16b^2 & 32ab^2 & 64a^2b^2 \\
1 & 3a & 9a^2 & 4b & 12ab & 36a^2b & 16b^2 & 48ab^2 & 144a^2b^2 \\
1 & 4a & 16a^2 & 4b & 16ab & 64a^2b & 16b^2 & 64ab^2 & 256a^2b^2
\end{bmatrix}
\begin{bmatrix}
c_{0,0} \\ c_{1,0} \\ c_{2,0} \\ c_{3,0} \\ c_{4,0} \\ c_{5,0} \\ c_{6,0} \\ c_{7,0} \\ c_{8,0}
\end{bmatrix}
=
\begin{bmatrix}
P(0,0,0) \\ P(0,a,0) \\ P(0,2a,0) \\ P(0,3a,0) \\ P(0,4a,0) \\ P(b,0,0) \\ P(b,a,0) \\ P(b,2a,0) \\ P(b,3a,0) \\ P(b,4a,0) \\ P(2b,0,0) \\ P(2b,a,0) \\ P(2b,2a,0) \\ P(2b,3a,0) \\ P(2b,4a,0) \\ P(3b,0,0) \\ P(3b,a,0) \\ P(3b,2a,0) \\ P(3b,3a,0) \\ P(3b,4a,0) \\ P(4b,0,0) \\ P(4b,a,0) \\ P(4b,2a,0) \\ P(4b,3a,0) \\ P(4b,4a,0)
\end{bmatrix}
$$

The inverse of **A** is found to be

$$
(A^{-1})^T = \begin{bmatrix}
\frac{961}{1225} & \frac{-837}{1225a} & \frac{31}{245a^2} & \frac{-837}{1225b} & \frac{729}{1225ab} & \frac{-27}{245a^2b} & \frac{31}{245b^2} & \frac{-27}{245ab^2} & \frac{1}{49a^2b^2} \\
\frac{279}{1225} & \frac{403}{2450a} & \frac{-31}{490a^2} & \frac{-243}{1225b} & \frac{-351}{2450ab} & \frac{27}{490a^2b} & \frac{9}{245b^2} & \frac{13}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{-93}{1225} & \frac{124}{245a} & \frac{-31}{245a^2} & \frac{81}{1225b} & \frac{-108}{245ab} & \frac{27}{245a^2b} & \frac{-3}{245b^2} & \frac{4}{49ab^2} & \frac{-1}{49a^2b^2} \\
\frac{-31}{245} & \frac{837}{2450a} & \frac{-31}{490a^2} & \frac{27}{245b} & \frac{-729}{2450ab} & \frac{27}{490a^2b} & \frac{-1}{49b^2} & \frac{27}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{93}{1225} & \frac{403}{1225a} & \frac{31}{245a^2} & \frac{-81}{1225b} & \frac{351}{1225ab} & \frac{-27}{245a^2b} & \frac{3}{245b^2} & \frac{-13}{245ab^2} & \frac{1}{49a^2b^2} \\
\frac{279}{1225} & \frac{-243}{1225a} & \frac{9}{245a^2} & \frac{403}{2450b} & \frac{-351}{2450ab} & \frac{13}{490a^2b} & \frac{-31}{490b^2} & \frac{27}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{81}{1225} & \frac{117}{2450a} & \frac{-9}{490a^2} & \frac{117}{2450b} & \frac{169}{4900ab} & \frac{-13}{980a^2b} & \frac{-9}{490b^2} & \frac{-13}{980ab^2} & \frac{1}{196a^2b^2} \\
\frac{-27}{1225} & \frac{36}{245a} & \frac{-9}{245a^2} & \frac{-39}{2450b} & \frac{26}{245ab} & \frac{-13}{490a^2b} & \frac{3}{490b^2} & \frac{-2}{49ab^2} & \frac{1}{98a^2b^2} \\
\frac{-9}{245} & \frac{243}{2450a} & \frac{-9}{490a^2} & \frac{-13}{490b} & \frac{351}{4900ab} & \frac{-13}{980a^2b} & \frac{1}{98b^2} & \frac{-27}{980ab^2} & \frac{1}{196a^2b^2} \\
\frac{27}{1225} & \frac{-117}{1225a} & \frac{9}{245a^2} & \frac{39}{2450b} & \frac{-169}{2450ab} & \frac{13}{490a^2b} & \frac{-3}{490b^2} & \frac{13}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{-93}{1225} & \frac{81}{1225a} & \frac{-3}{245a^2} & \frac{124}{245b} & \frac{-108}{245ab} & \frac{4}{49a^2b} & \frac{-31}{245b^2} & \frac{27}{245ab^2} & \frac{-1}{49a^2b^2} \\
\frac{-27}{1225} & \frac{-39}{2450a} & \frac{3}{490a^2} & \frac{36}{245b} & \frac{26}{245ab} & \frac{-2}{49a^2b} & \frac{-9}{245b^2} & \frac{-13}{490ab^2} & \frac{1}{98a^2b^2} \\
\frac{9}{1225} & \frac{-12}{245a} & \frac{3}{245a^2} & \frac{-12}{245b} & \frac{16}{49ab} & \frac{-4}{49a^2b} & \frac{3}{245b^2} & \frac{-4}{49ab^2} & \frac{1}{49a^2b^2} \\
\frac{3}{245} & \frac{-81}{2450a} & \frac{3}{490a^2} & \frac{-4}{49b} & \frac{54}{245ab} & \frac{-2}{49a^2b} & \frac{1}{49b^2} & \frac{-27}{490ab^2} & \frac{1}{98a^2b^2} \\
\frac{-9}{1225} & \frac{39}{1225a} & \frac{-3}{245a^2} & \frac{12}{245b} & \frac{-52}{245ab} & \frac{4}{49a^2b} & \frac{-3}{245b^2} & \frac{13}{245ab^2} & \frac{-1}{49a^2b^2} \\
\frac{-31}{245} & \frac{27}{245a} & \frac{-1}{49a^2} & \frac{837}{2450b} & \frac{-729}{2450ab} & \frac{27}{490a^2b} & \frac{-31}{490b^2} & \frac{27}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{-9}{245} & \frac{-13}{490a} & \frac{1}{98a^2} & \frac{243}{2450b} & \frac{351}{4900ab} & \frac{-27}{980a^2b} & \frac{-9}{490b^2} & \frac{-13}{980ab^2} & \frac{1}{196a^2b^2} \\
\frac{3}{245} & \frac{-4}{49a} & \frac{1}{49a^2} & \frac{-81}{2450b} & \frac{54}{245ab} & \frac{-27}{490a^2b} & \frac{3}{490b^2} & \frac{-2}{49ab^2} & \frac{1}{98a^2b^2} \\
\frac{1}{49} & \frac{-27}{490a} & \frac{1}{98a^2} & \frac{-27}{490b} & \frac{729}{4900ab} & \frac{-27}{980a^2b} & \frac{1}{98b^2} & \frac{-27}{980ab^2} & \frac{1}{196a^2b^2} \\
\frac{-3}{245} & \frac{13}{245a} & \frac{-1}{49a^2} & \frac{81}{2450b} & \frac{-351}{2450ab} & \frac{27}{490a^2b} & \frac{-3}{490b^2} & \frac{13}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{93}{1225} & \frac{-81}{1225a} & \frac{3}{245a^2} & \frac{-403}{1225b} & \frac{351}{1225ab} & \frac{-13}{245a^2b} & \frac{31}{245b^2} & \frac{-27}{245ab^2} & \frac{1}{49a^2b^2} \\
\frac{27}{1225} & \frac{39}{2450a} & \frac{-3}{490a^2} & \frac{-117}{1225b} & \frac{-169}{2450ab} & \frac{13}{490a^2b} & \frac{9}{245b^2} & \frac{13}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{-9}{1225} & \frac{12}{245a} & \frac{-3}{245a^2} & \frac{39}{1225b} & \frac{-52}{245ab} & \frac{13}{245a^2b} & \frac{-3}{245b^2} & \frac{4}{49ab^2} & \frac{-1}{49a^2b^2} \\
\frac{-3}{245} & \frac{81}{2450a} & \frac{-3}{490a^2} & \frac{13}{245b} & \frac{-351}{2450ab} & \frac{13}{490a^2b} & \frac{-1}{49b^2} & \frac{27}{490ab^2} & \frac{-1}{98a^2b^2} \\
\frac{9}{1225} & \frac{-39}{1225a} & \frac{3}{245a^2} & \frac{-39}{1225b} & \frac{169}{1225ab} & \frac{-13}{245a^2b} & \frac{3}{245b^2} & \frac{-13}{245ab^2} & \frac{1}{49a^2b^2}
\end{bmatrix}
$$

Now $\vec{\mathbf{C}}$ is calculated as $\vec{\mathbf{C}} = \vec{\mathbf{A}}^{-1}\vec{\mathbf{B}}$, and is used for finding a floating-point index pointing into the data array. This floating-point index is either rounded, so that the nearest point is chosen, or is used for a linear interpolation between the two closest data points, so that a data value is found.

Two options are available for finding the coefficients $a_0 \ldots a_{26}$. The first method is to go through the same process as shown but for the full problem. That process would involve enlarging $\vec{\mathbf{C}}$ to hold the 27 coefficients $(a_{0\ldots26})$; enlarging $\vec{\mathbf{B}}$ to hold the ideal values for more than one position in the aperture (say five positions); and enlarging $\vec{\mathbf{A}}$ to match.

The second method is to solve for vector $\vec{\mathbf{C}}$ (as shown) at a number of positions in the aperture. Then construct a polynomial in $j$ for each coefficient. This solution does not involve inverting the large **A** matrix but will result in a less than optimum solution in the least-squares sense.

One advantage of this second method is that it lends itself to correcting for airplane motion. Since **A** is independent of the airplane position, it is inverted once. The matrix multiplication to find the index polynomial coefficients can be applied at every aperture position or short subaperture if desired. The algorithm thus allows real-time UWB motion compensation.

### 6.2.3 *Fast Polynomial Calculation*

Now that we have a polynomial to find the index, we need a fast way to compute the polynomial. Directly calculating an $N^{\text{th}}$ degree polynomial usually requires $N$ adds and $N$ multiplies. If, however, a sequence of solutions is desired with the variable changed in fixed increments (the case we have here), more efficient means are available. An algorithm by Nuttall [8] describes a method that can be applied here, to calculate the index recursively without the multiplications. The procedure is as follows:

1. Let $X(i) = q_0 + q_1 i + q_2 i^2$ be the polynomial to be solved, where $i = 0, 1, 2, 3\ldots$.

2. Let $X_1(i) = X(i) - X(i-1) = q_1 - q_2 + 2q_2 i$. Observe that $X_1(i) - X_1(i-1) = 2q_2$.

3. Therefore a recursion can be set up where

$$X_1(i) = X_1(i-1) + 2q_2, \text{ and}$$
$$X(i) = X_1(i-1) + X_1(i)$$

The starting values for the recursion are

$$X(0) = q_0 \text{ and}$$
$$X_1(0) = q_1 - q_2 + 2q_2 i = q_1 - q_2.$$

The same derivation can be applied to an arbitrary $N^{th}$ degree polynomial to produce a recursive formula that requires $N$ adds per step. The technique can also be expanded to multiple dimensions by nesting. Appendix B lists all routines used in the fast focusing algorithm. Subroutine poly2 in section B-5 of appendix B uses this technique directly.

### 6.2.4 *Coefficient Generator Program*

Figure 9 shows how the image is broken down, along with the names used in the subroutines.

The basic flow chart is shown in figure 10. In figure 10, the $Z_p$'s are the $a_0$ to $a_{26}$ coefficients. Each time the inner loop goes through all the $n$'s, the $Z_p$'s are calculated for the box (defined by $h$ and $m$) and the subaperture

**Figure 9. Image partitioning and notation.**

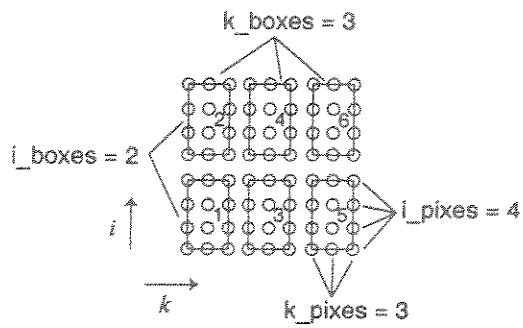
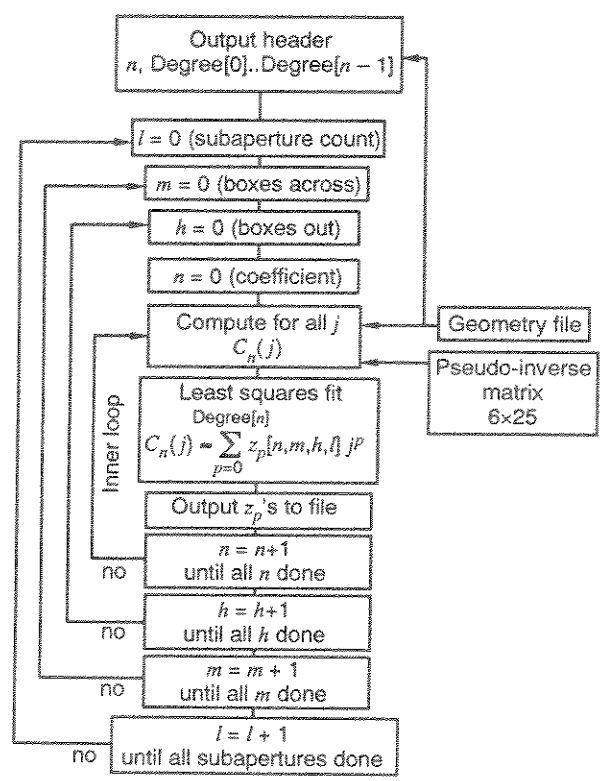



**Figure 10. Coefficient generator flow chart.**



(defined by $l$). The flow chart shows that the polynomial degree for the $C_n$ is not always the same. The degree was optimized based on the errors caused when a lower degree was tried. Appendix A provides a complete listing of the program that generates the focusing coefficients from an input file with the geometry.

## 6.3 Fast Focusing Algorithm

The fast focusing algorithm simply applies the fast polynomial solution technique to find the indexing and does the signal summation into all the pixels in the image. As outlined above, the index calculation is done in three nested loops. To aid in explanation, we provide displays of subroutines Inner_Loop, Middle_Loop, and Outer_Loop as pseudo-coded examples of how the focusing routine is written using the recursive technique with nesting. A second-degree polynomial is used throughout for illustration purposes. These subroutines assume that the image is broken into a number of boxes where each box has its own set of coefficients. Inner_Loop (display 1) is the simplest subroutine and follows the derivation of the recursive formula directly. It essentially increments $i$ to focus a single line in a box.

**Display 1. Subroutine Inner_Loop**

| Code | Comments |
|---|---|
| Inner_Loop (f, s, s_max, q, i_pixes) | Inner loop subroutine |
| float *f; | Pointer to first pixel in a line; f(i_start,k) |
| float *s; | Pointer to signal-data vector from $j$th aperture position |
| float *q; | Pointer to coefficients vector for index calculation |
| int s_max; | $s$ range is s[0]...s[s_max] s_max. |
| int i_pixes; | Size of the box (pixels) along $i$ axis is i_pixes. |
| f_stop_i = f + i_pixes–1;<br>T = q[2]+q[2];<br>R1 = q[1]–q[2];<br>R0 = q[0]; | Set up initial conditions. |
| If R0 < –0.5 then repeat<br>{f++;<br>R1 = R1+T;<br>R0 = R0+R1};<br>until R0 >–0.5};<br>If (f > f_stop_i) then return; | Perform clipping when index < 0 (before beginning of actual data). |
| index = Round(R0);<br>*f = *f + *(s+index); | Perform first summation. |
| repeat | Begin loop for a line in box. |
| {f++; | Increment pointer to next pixel. |
| R1 = R1+T; | Increment index R0 = R0+R1. |
| index = Round(R0);<br>if index>s_max return; | Clip when index is beyond actual data record length. |
| *f = *f +;*(s+index) | Sum into pixel the new data. |
| until (f = f_stop);<br>return; | END Inner_Loop |

Nesting of the recursive approach means that the coefficients $q_0$, $q_1$, and $q_2$ are each formed by polynomials in $k$. Each polynomial is incremented recursively in the subroutine Middle_Loop (display 2). Thus, Middle_Loop sums into a single box the data from a single aperture position.

**Display 2. Subroutine Middle_Loop**

| Code | Comments |
|---|---|
| Middle_Loop(f, s, c, s_max, i_pixes, k_inc, box_offset); | Middle loop subroutine |
| float *f; | Pointer to first pixel of box |
| float *s; | Pointer to *j*th data vector |
| float *c; | Pointer to coefficients vector for index calculation |
| int s_max; | *s* range is s[0]...s[s_max] |
| int k_inc; | F[i,k] = *(f + i + k * k_inc) so k_inc is total range line length. |
| int i_pixes; | Size of the box (pixels) along *i* axis. |
| int box_offset; | Number to add to *f* to move pointer to last line in box. |
| f_last = f + box_offset; | f_last = address of the first pixel in last line of box. |
| q0_T = c[2]+c[2];<br>q0_R1 = c[1]–c[2];<br>q[0] = c[0]; | Set up initial conditions for $q_0$. |
| q1_T = c[5]+c[5];<br>q1_R1 = c[4]–c[5];<br>q[1] = c[3]; | Set up initial conditions for $q_1$. |
| q2_T = c[8]+c[8];<br>q2_R1 = c[7]–c[8];<br>q[2] = c[6]; | Set up initial conditions for $q_2$. |
| Call Inner_Loop(f,s,q..); | Perform summation on first line. |
| repeat | Start loop for rest of lines. |
| f = f + k_inc; | Increment pointer to next line in box. |
| q0_R1 = q0_R1+q0_T;<br>q[0] = q[0]+q0_R1; | Iterate $q_0$. |
| q1_R1 = q1_R1+q1_T;<br>q[1] = q[1]+q1_R1 | Iterate $q_1$. |
| q2_R1 = q2_R1+q2_T;<br>q[2] = q[2]+q2_R1; | Iterate $q_2$. |
| Call Inner_Loop(..); | Perform summation on current line in box. |
| Until (f = f_last);<br>Return; | Loop until all lines done. |

Again, nesting of the recursive approach means that the coefficients $c_0 \ldots c_8$ are each formed by polynomials in $j$. Each polynomial is incremented recursively in the subroutine Outer_Loop (display 3). Outer_Loop thus sums into each box the data from multiple aperture positions. Since it is desirable to read the signal data only once, Outer_Loop will both call Middle_Loop for each box and increment the coefficients for each box separately as it increments $j$ across the aperture.

**Display 3. Subroutine Outer_Loop**

| Code | Comments |
|---|---|
| Outer_Loop(f,a,s_max,k_boxes,i_boxes, k_pixes,i_pixes,j_beg, j_end) | Outer loop subroutine |
| float *fo; | Pointer to first pixel in entire image; all pixels set to zero. |
| float *a; | Pointer to array of coefficient vectors for index calculation; one vector per box. |
| int s_max; | $s$ has range of s[0] to s[s_max] |
| int | Number of boxes in $k$ axis k_boxes; |
| int | Number of boxes in $i$ axis i_boxes; |
| int k_pixes; | Size of box (pixels) in $k$ axis |
| int | Size of box (pixels) in $i$ axis i_pixes; |
| int j_beg; | Start of data vectors to be processed |
| int j_end; | Stop of data vectors to be processed |
| k_inc = i_pixes * i_boxes; | (F(i,k)) = *(f+i+k*k_inc) |
| box_offset = k_inc * (k_pixes–1); | Used by Middle_Loop |
| box_inc_k = i_pixes + box_offset; | box_inc_i = i_pixes; |
| j = j_start;<br>read s[0..s_max] for j th position; | Read signal data for first aperture position. |
| f = fo – (box_inc_k+box_inc_i);<br>box = –2;<br>for k_box = 0 to k_boxes–1;<br>box = box + 1;<br>f = f + box_inc_k;<br>for i_box = 0 to i_boxes–1;<br>box = box + 1;<br>f = f + box_inc_i; | Set up to loop through all boxes. |
| c0_T[box] = a[2,box]+a[2,box];<br>c0_R1[box] = a[1,box]–a[2,box];<br>c[0,box] = a[0,box]; | Initial condition for $c_0$ for current box. |
| c1_T[box] = a[5,box]+a[5,box];<br>c1_R1[box] = a[4,box]–a[5,box];<br>c1 c[1,box] = a[3,box]; | Initial condition for $c_1$ for current box. |
| -<br>-<br>- | Initial condition for $c_2 \ldots c_7$ for current box. |
| c8_T[box] = a[27,box]+a[27,box];<br>c8_R1[box] = a[26,box]–a[27,box];<br>c[8,box] = a[25,box]; | Initial condition for $c_8$ for current box. |
| Call Middle_Loop(f,s,c[box],...); | First summation for current box. |

**Display 3. Subroutine Outer_Loop (cont'd)**

| Code | Comments |
|---|---|
| END (for i_box);<br>END (for k_box); | Loop through all boxes. |
| repeat | Loop through all aperture positions. |
| j = j+1; | Increment aperture. |
| read s[0..s_max]; | Position read data for that aperture position |
| f = fo – (box_inc_k+box_inc_i); | Initialize *f*. |
| box = –2;<br>for k_box = 0 to k_boxes–1;<br>box = box + 1;<br>f = f + box_inc_k;<br>for i_box = 0 to i_boxes–1;<br>box = box + 1;<br>f = f + box_inc_i; | Set up to loop through all boxes. |
| c0_R1[box] = c0_R1[box]+c0_T[box];<br>c[0,box] = c[0,box]+c0_R1[box]; | Iterate $c_0$ for current box. |
| c1_R1[box] = c1_R1[box]+c1_T[box];<br>c[1,box] = c[1,box]+c1_R1[box]; | Iterate $c_1$ for current box. |
| ... | Iterate $c_2 ... c_7$ for current box. |
| c8_R1[box] = c8_R1[box]+c8_T[box];<br>c[8,box] = c[8,box]+c8_R1[box]; | Iterate $c_8$ for current box. |
| Call Middle_Loop(f,s,c[box],..); | Sum into current box. |
| END (for i_box);<br>END (for k_box); | Loop until all boxes done. |
| Until j = j_stop;<br>Return (END Outer_Loop) | Loop until all aperture positions are done. |

## 6.4 *Considerations for Fast Focusing Polynomial Order Selection*

A study was conducted to determine the effect of polynomial order on the image size and aperture length. The worst-case position in the aperture ($j$ = 3 in fig. 3, p 17) and the worst-case area in the image ($(i,k)$ = (237,3) in fig. 3) were chosen for the analysis. The sample rate was 4 Gs/s with a record length (s_max) of 4096 points. The elevation was 60 ft; otherwise the geometry was as in figure 3, with an aperture $L_s$ = 384 ft, squint angle $\theta = 85°$, and slant range $R$ = 550 ft. If the error in the floating-point index is restricted to ±0.5, then this geometry produces the following results:

| Polynomial degree | Maximum i_pixes | Maximum k_pixes |
|---|---|---|
| 1 | 228 | 27 |
| 2 | 964 | 105 |
| 3 | 2191 | 315 |
| 4 | — | 657 |

By using the nested recursive technique as shown in the example subroutines, we compute the computational load as shown in table 2.

Grouping terms, we get

$$L = \text{(inner loop adds)} + \text{(middle loop adds)} + \text{(outer loop adds)}$$

$$= IKJB(i' + 1) + KJB(i' + 1)k' + JB(i' + 1)(k' + 1)j' .$$

From this equation, it is clear that a low-order polynomial is crucial in the inner loop. Conversely, the order of the polynomial in the outer loop can be high with little impact on the overall speed. Table 3 shows the computational load for various configurations with $j' = 3$ and $J = 2304$.

**Table 2. Computational load.**

| Symbol | Definition |
|---|---|
| $i'$ | Polynomial order for $i$ index |
| $k'$ | Polynomial order for $k$ index |
| $j'$ | Polynomial order for $j$ index |
| $J$ | Number of points in the aperture |
| $I$ | Number of range bins (pixels) in box (i_pixes) |
| $K$ | Number of azimuth bins (pixels) in box (k_pixes) |
| $B$ | Number of boxes |
| $L_0$ | $I(i' + 1)$; adds per Inner_Loop call |
| $L_1$ | $K[L_0 + (i' + 1)k']$; adds per Middle_Loop call |
| $L$ | $J[L_1 + (i' + 1)(k' + 1)j']$; adds per Outer_Loop (adds per image) |

**Table 3. Matrix showing computational load versus partitioning for $j' = 3$ and $J = 2304$.**

| | | | $k'$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|
| | | | $K$ | 27 | 103 | 256 | 512 |
| | | | Boxes across | 19 | 5 | 2 | 1 |
| | | | Total pixels across | 513 | 515 | 512 | 512 |
| $i'$ | $I$ | Boxes down | Total pixels down | | | | |
| 1 | 227 | 18 | 4086 | $L = 9.711$<br>$S = 21{,}888$<br>$B = 342$ | $L = 9.786$<br>$S = 8640$<br>$B = 90$ | $L = 9.769$<br>$S = 4608$<br>$B = 36$ | $L = 9.810$<br>$S = 2880$<br>$B = 18$ |
| 2 | 817 | 5 | 4085 | $L = 14.51$<br>$S = 9120$<br>$B = 95$ | $L = 14.58$<br>$S = 3600$<br>$B = 25$ | $L = 14.51$<br>$S = 1920$<br>$B = 10$ | $L = 14.53$<br>$S = 1200$<br>$B = 5$ |
| 3 | 2043 | 2 | 4086 | $L = 19.33$<br>$S = 4864$<br>$B = 38$ | $L = 19.41$<br>$S = 1920$<br>$B = 10$ | $L = 19.31$<br>$S = 1024$<br>$B = 4$ | $L = 19.32$<br>$S = 640$<br>$B = 2$ |

*$L$ = total number of adds in giga-adds.*
*$B$ = total number of boxes in the image.*
*$S$ = bytes of storage needed for the coefficient tables = $(i' + 1)(k' + 1)(j' + 1)B$(4 bytes/word).*

## 6.5 Implementation Notes

The algorithm developed here requires no multiplication except in the preprocessing (interpolation) stage. There already exist DSP (digital signal processor) chips whose architecture is ideal for the FIR interpolation task. The bulk of the computations, however, occur in the summing and index calculation stage. Two facts make the summing algorithm attractive. First, adders take considerably less area to implement in a VLSI (very-large-scale integration) chip than multipliers. Second, parallel operation is extremely simple; the image can simply be broken into boxes with a separate processor working independently on each box. In this case, only the coefficient table would be different between processors, and a broadcast mode would be needed to pass the signal data to each processor. Therefore, design of a custom LSI chip appears practical and could result in real-time speeds.

If an off-the-shelf DSP chip with a parallel adder and multiplier is used to perform this algorithm, then other options become available. For example, the multiplier can be used in the summing algorithm to do interpolation on the indexing rather than rounding. The multiplication required can be done during other add cycles so that it takes zero time. Another possibility is to calculate the index polynomial directly instead of recursively. Finally, since all the operations can be fixed point, the address-generator ALU (arithmetic logic unit) can sometimes be used in parallel with the floating-point units.

# 7. Beam Patterns and Sidelobe Structure

To develop an intuitive understanding for the beam shape, we assembled 16 figures to display, in the time domain, the main-lobe and sidelobe beam patterns. We simulated a resonant target by generating a data record for every position in the aperture. The simulated scene was a single-point target. The target bearing (see fig. 3) was squinted 15° off broadside ($\theta = 105°$). Range to the target $R$ was 750 ft, the aperture length $L_s$ was 385 ft, and the aperture height was 60 ft. The point target had an impulse response of

$$s(t) = \begin{cases} \sin(2\pi ft)[0.5 + 0.5\cos(4\pi ft)] & \text{for } 0 < t < \dfrac{1}{4f} \\ \sin(2\pi ft)\exp\left[\left(\dfrac{1}{4f - t}\right)0.23f\right] & \text{for } t \geq \dfrac{1}{4f} \end{cases} \tag{15}$$

The data simulated a 2-GHz sample rate. A record length of 2048 samples was made for each aperture position. Each record was preprocessed with a times-8 interpolator, producing 16K records. Interpolation was done by the standard FIR filter method. A 255-tap Parks-McClellan low-pass filter with a 950-MHz cutoff was used to do the interpolation. Equation (7) (with $n$ = 0) was used to produce the focused beams. A Hilbert transform was used to obtain the magnitude that is plotted in the figures. Plots were made on a decibel scale. Four 3-D plots were made, showing two viewing angles (on axis and above axis) and two aperture weighting functions (Hamming and rectangular) for each of four targets with different resonant frequencies (50, 200, 400, and 900 MHz). These are shown in figures 11 to 26.

The on-axis plots were made so that amplitude values could be easily read off the plots. The above-axis plots were made to reveal the sidelobe structure and the ring-down time of the target. The axis labels were shifted so that 0 range was at the point target and 0° was the bearing centered on the target. Notice that all plots are 3-D, even the on-axis plots. The various horizontal lines in the on-axis plots are the beam pattern on the $i^{th}$ range bin. These horizontal lines are identical to those shown in the above-axis plots; they are just being viewed "end on."

These figures are unique in that they show how the sidelobes spread in time as one moves off the main beam. Because of the sharp rise time of the target echo, the contribution from the near and far ends of the aperture can be seen as the early and late peaks in the sidelobe structure. Hamming weighting was applied to the array so that one can see the effect of weighting on the sidelobe structure. It is interesting to note that although the peak sidelobe levels drop only marginally, the average or integrated sidelobe levels are significantly lower when tapered aperture weighting is used. The weighting also reduces the near and far peaks in the sidelobes since the weighting reduces the contribution of the array ends.

Since the antenna is a linear system, it is no surprise that the beam pattern behaves basically as classic antenna theory predicts. The aperture beam pattern is a function of (1) how long the aperture is relative to wavelength, and (2) what kind of weighting is used to sum the points in the aperture. If uniform aperture weighting is used (a straight summation), then the typical $\sin(\beta\psi)/(\beta\psi)$ pattern occurs, where $\psi$ is the angle off the main beam, and $\beta \propto L_S/\lambda$. $\beta$ establishes the width of the main beam. As the wavelength gets small, or as the aperture gets long, the beam gets narrow.

**Figure 11. 50-MHz point reflector, equal weighting, end view.**

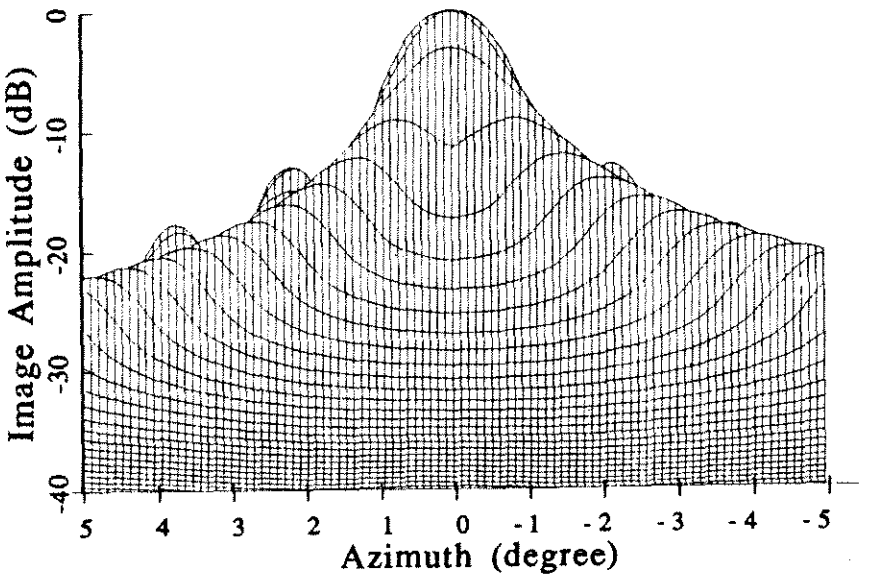



**Figure 12. 50-MHz point reflector, Hamming weighting, end view.**

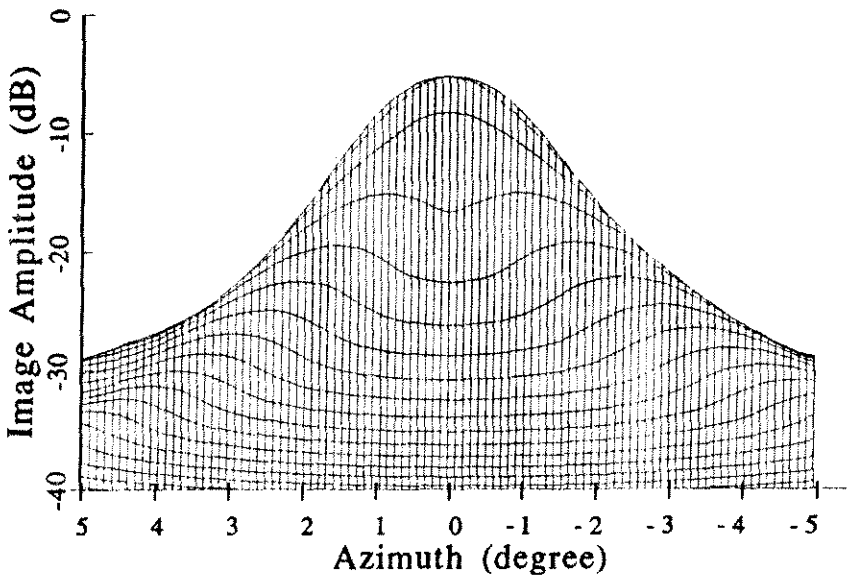


Figure 13. 50-MHz point reflector, equal weighting, above axis.

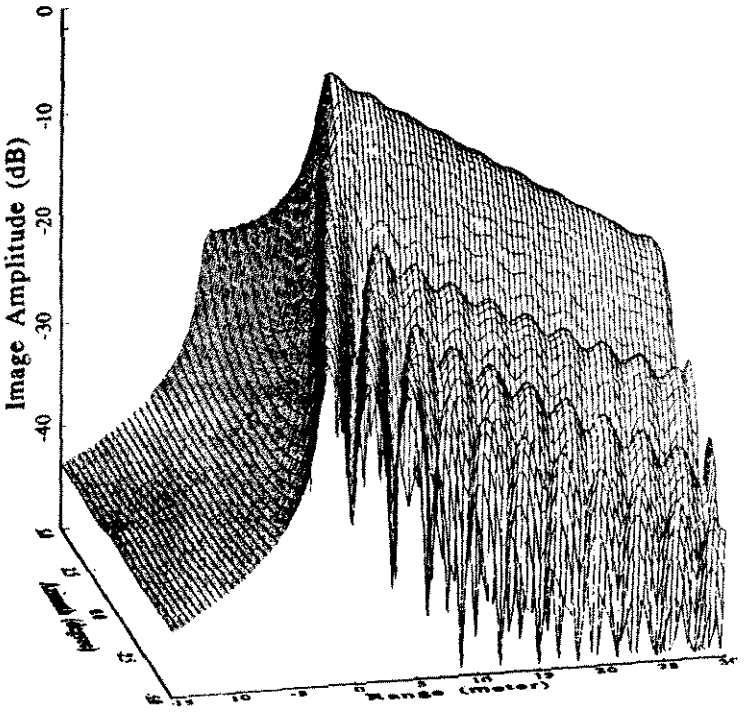



Figure 14. 50-MHz point reflector, Hamming weighting, above axis.

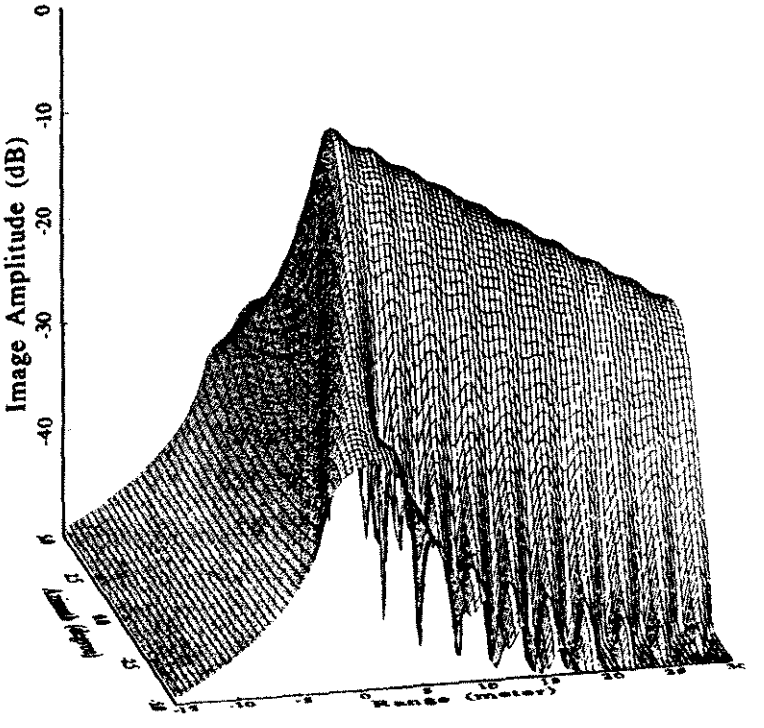


Figure 15. 200-MHz point reflector, equal weighting, end view.

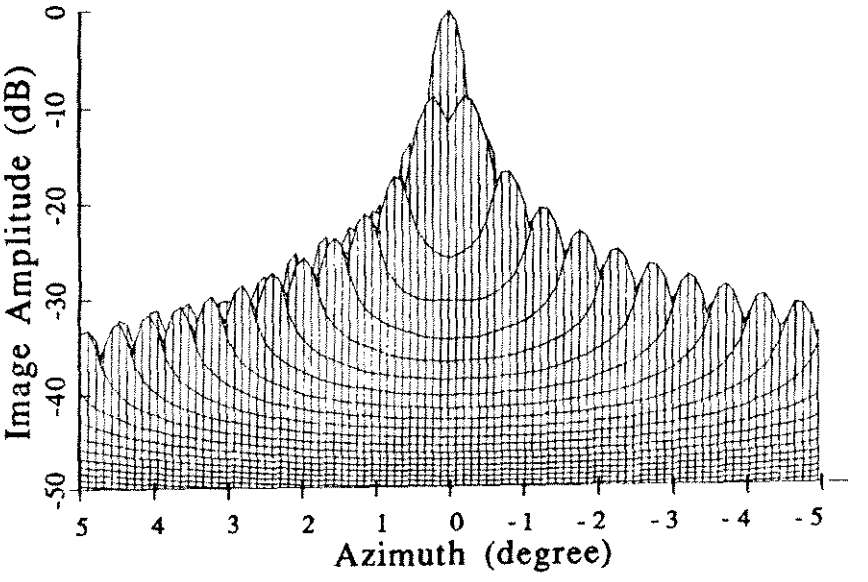



Figure 16. 200-MHz point reflector, Hamming weighting, end view.

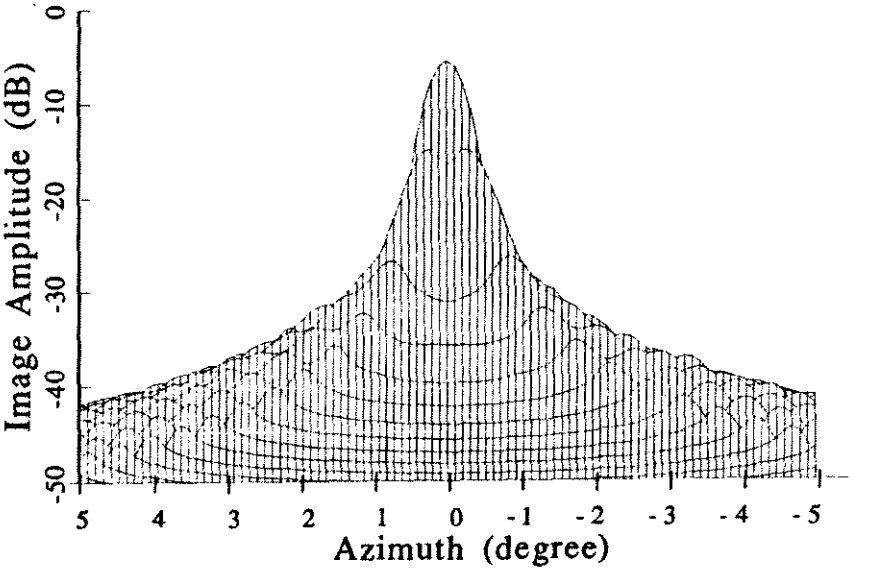


Figure 17. 200-MHz point reflector, equal weighting, above axis.

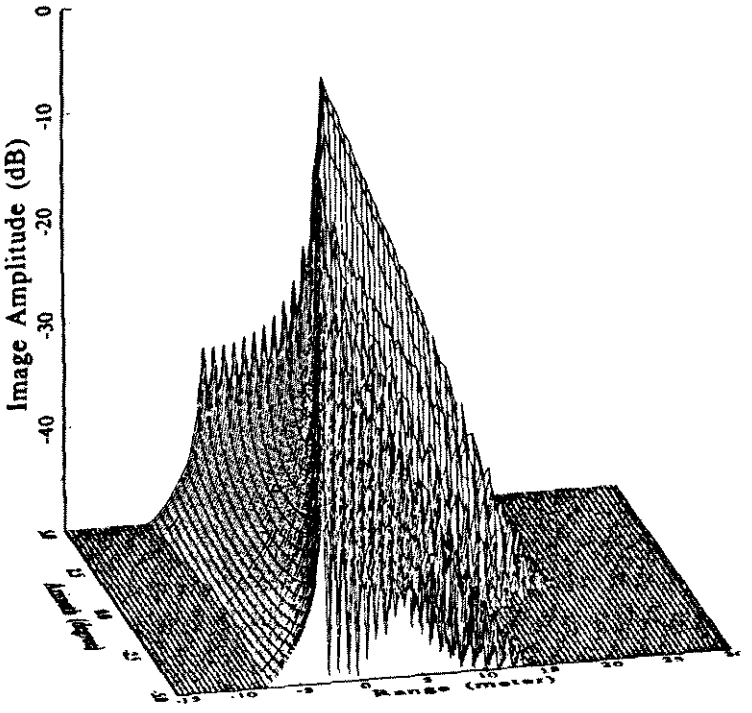



Figure 18. 200-MHz point reflector, Hamming weighting, above axis.

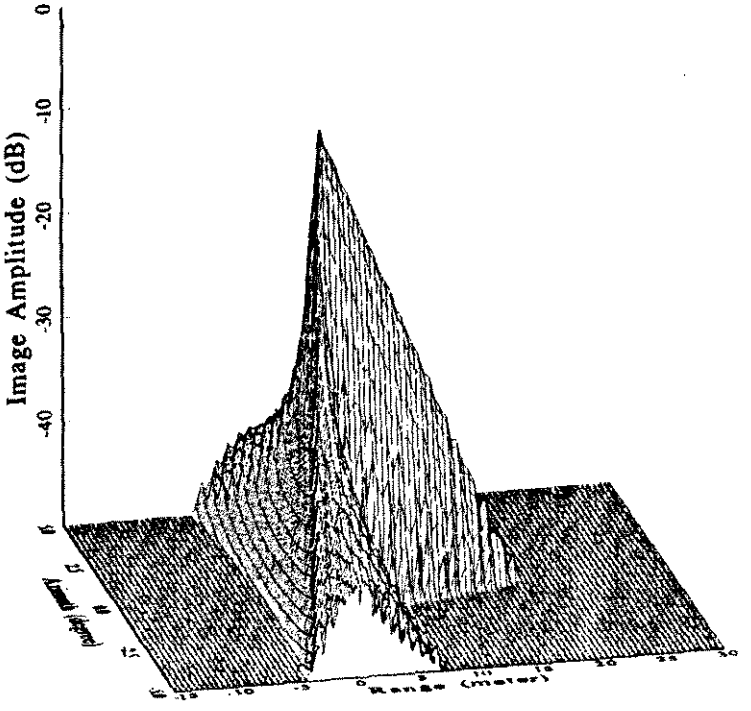


Figure 19. 400-MHz point reflector, Hamming weighting, end view.

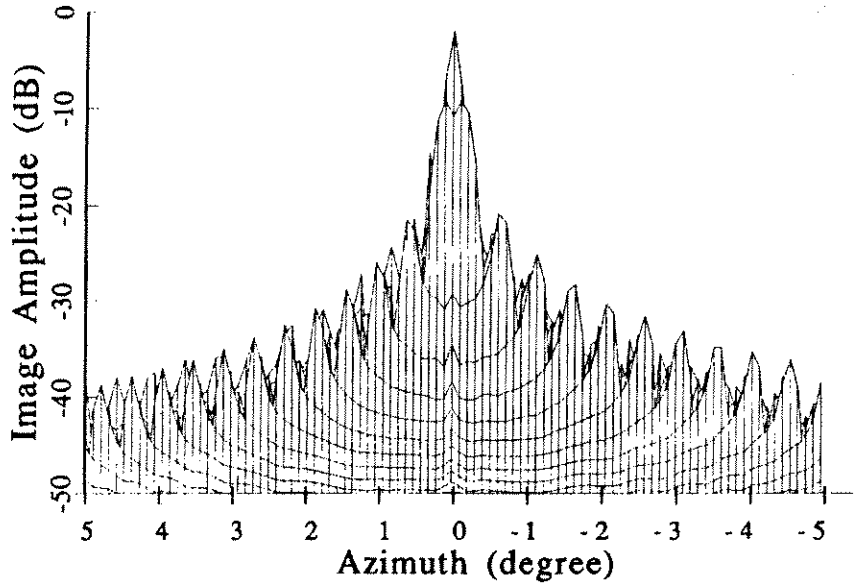



Figure 20. 400-MHz point reflector, Hamming weighting, end view.

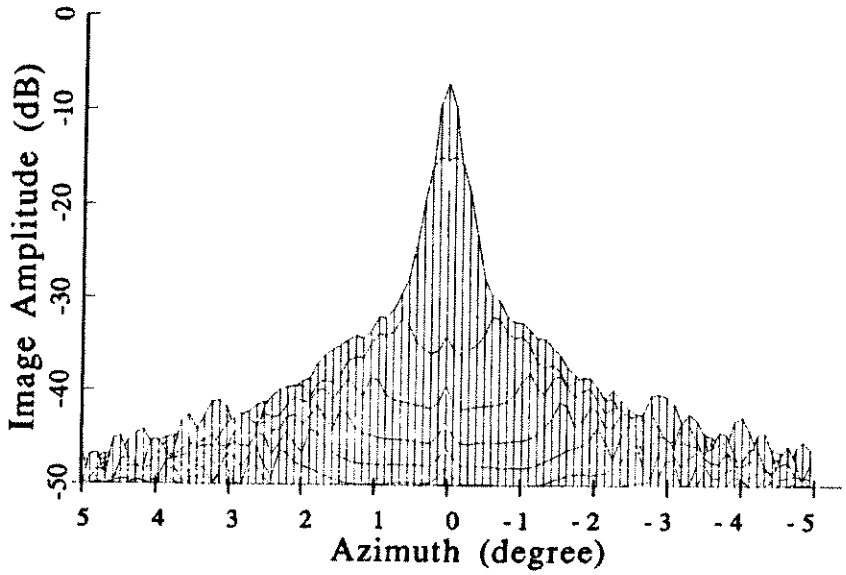


Figure 21. 400-MHz point reflector, equal weighting, above axis.

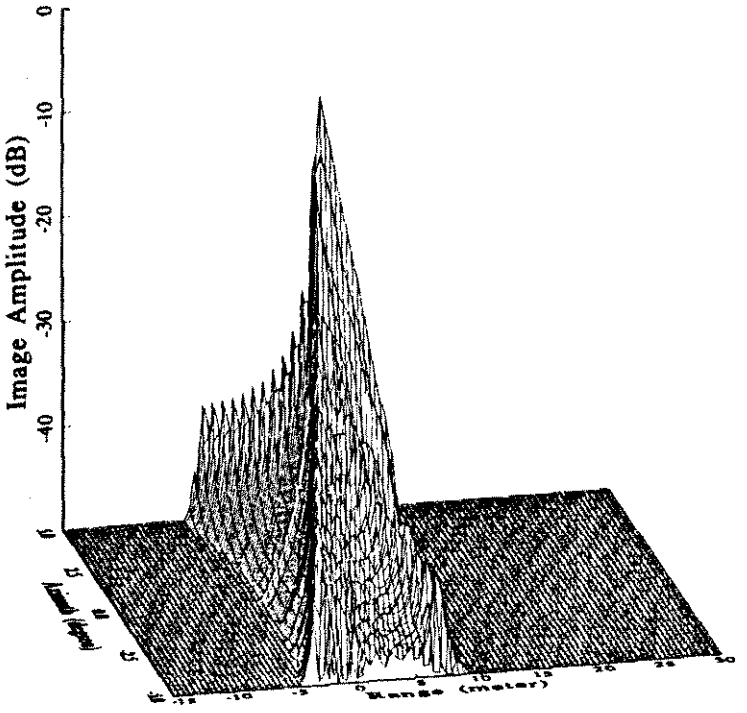



Figure 22. 400-MHz point reflector, Hamming weighting, above axis.

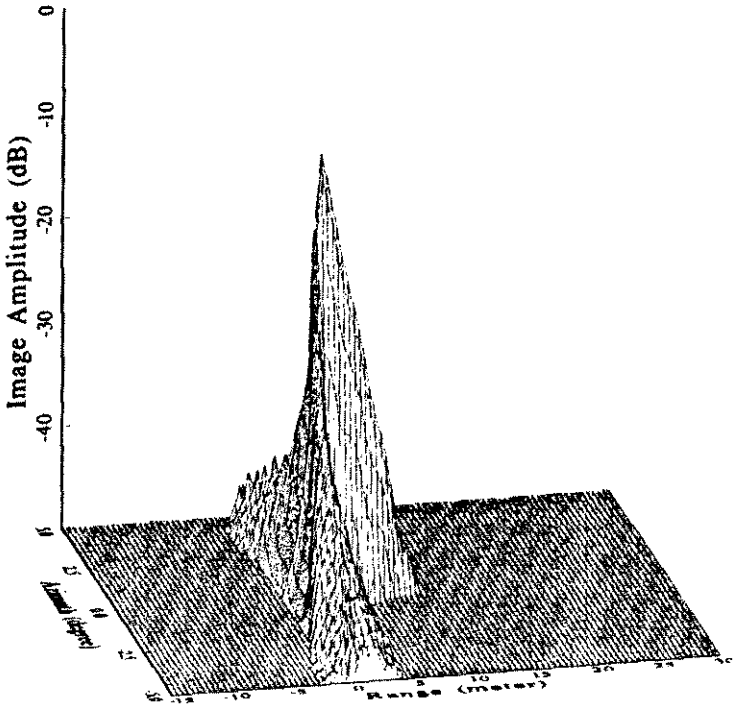


Figure 23. 900-MHz point reflector, equal weighting, end view.

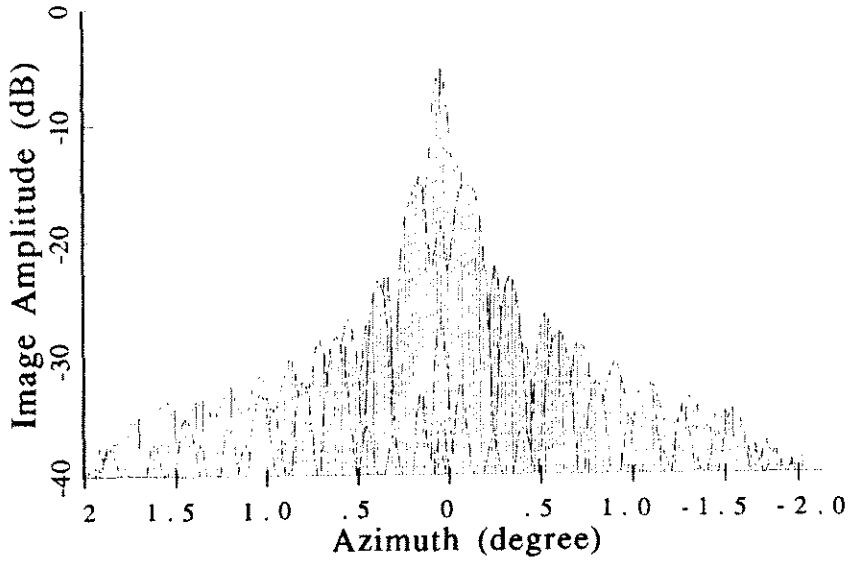



Figure 24. 900-MHz point reflector, Hamming weighting, end view.

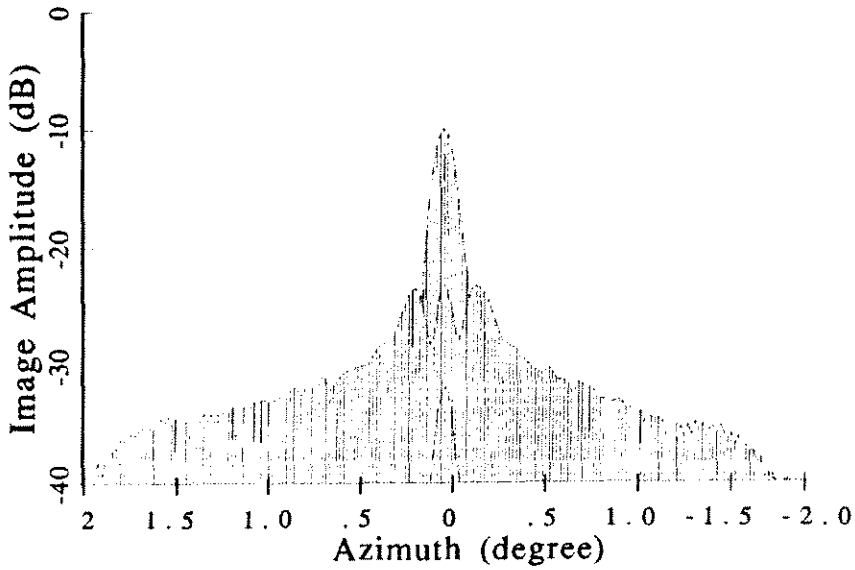


Figure 25. 900-MHz point reflector, equal weighting, above axis.

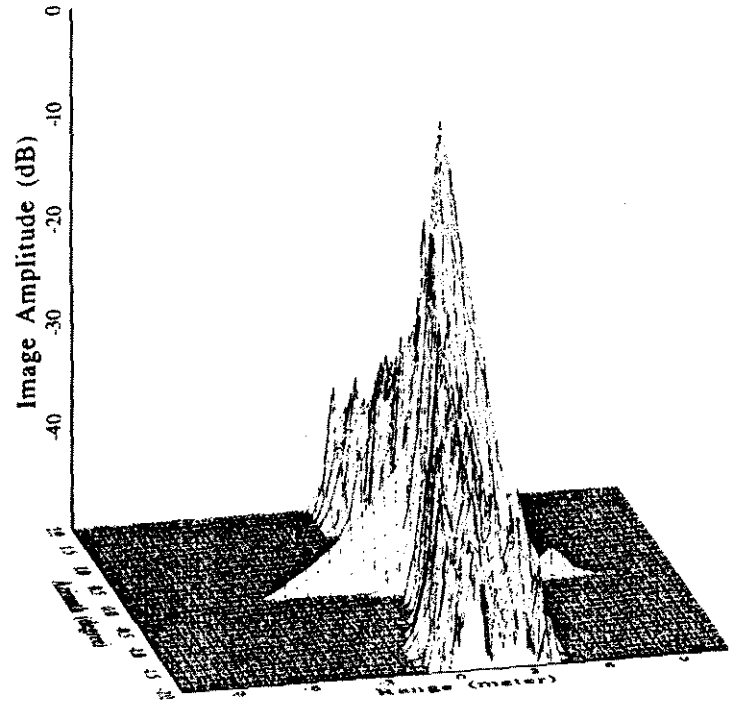
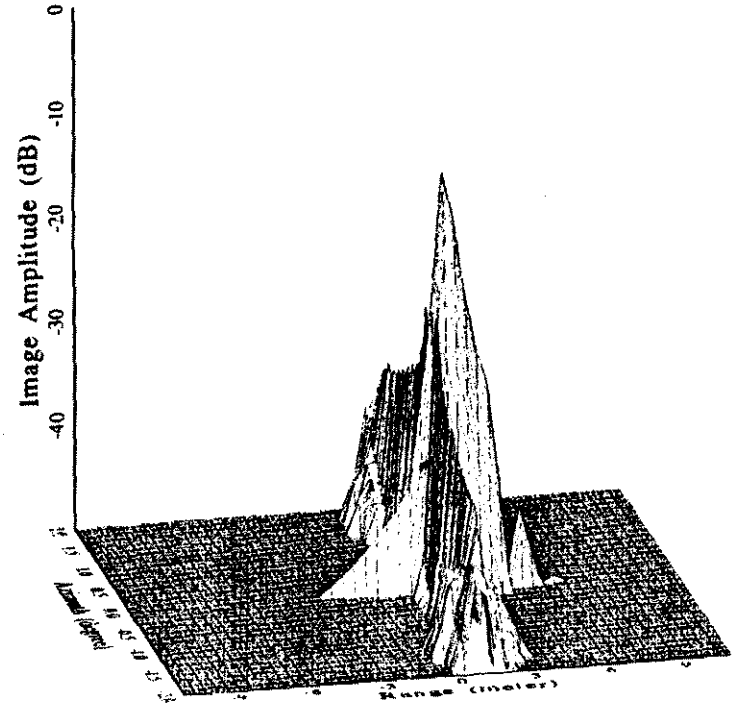



Figure 26. 900-MHz point reflector, Hamming weighting, above axis.



# 8. Summary

This report identifies a frequency-independent processing algorithm to get a perfectly focused impulse response from any object at any position from an ultra-wide-bandwidth synthetic aperture radar. The report also presents an approximation that reduces the computational complexity of the algorithm. An error analysis of this approximation demonstrates its applicability to focus even high-$Q$ objects in the near field of an aperture. An implementation that is both computationally efficient and applicable to real-time motion compensation is also presented. Finally, plots are presented demonstrating the capability of the algorithm to focus ringing targets over an 18-to-1 bandwidth.

# References


1. William M. Brown and Ronald J. Fredricks, *Range-Doppler Imaging with Motion through Resolution Cells*, IEEE Trans. Aerosp. Electron. Syst. **AES-5**, No. 1 (January 1969), 98–102.

2. J. L. Bauck and W. K. Jenkins, *Tomographic Processing of Spotlight-Mode Synthetic Aperture Radar Signals with Compensation for Wavefront Curvature*, IEEE Int. Conf. Acoust. Speech Signal Process. (11–14 April 1988).

3. Edwin D. Banta, *Limitations on SAR Image Area Due to Motion Through Resolution Cells*, Correspondence, IEEE Trans. Aerosp. Electron. Syst. **AES-22**, No. 6 (November 1986), 799–803.

4. D. Mensa and G. Heidbreder, *Bistatic Synthetic-Aperture Radar Imaging of Rotating Objects*, IEEE Trans. Aerosp. Electron. Syst. **AES-18**, No. 4 (July 1982), 423–431.

5. Dale A. Ausherman, Adam Kozma, Jack L. Walker, Harrison M. Jones, and Enrico C. Poggio, *Developments in Radar Imaging,* IEEE Trans. Aerosp. Electron. Syst. **AES-20**, No. 5 (July 1984), 363–400.

6. J. L. Walker, *Range-Doppler Imaging of Rotating Objects,* IEEE Trans. Aerosp. Electron. Syst. **AES-16** (January 1980), 23–52.

7. J. A. Landt, E. K. Miller, and M. Van Blaricum, *WT-MBA/LLL1B: A Computer Program for the Time-Domain Electromagnetic Response of Thin-Wire Structures,* Lawrence Livermore Laboratory (6 May 1974).

8. Albert H. Nuttall, *Efficient Evaluation of Polynomials and Exponentials of Polynomials for Equispaced Arguments*, IEEE Trans. Acoust. Speech Signal Process. **ASSP-35**, No. 10 (October 1987), 1486–1487.

# Appendix A. Coefficient Generator Program

## Appendix A

### Contents

page

A-1. Main Program to Calculate Coefficients (main.c) ..... 45
A-2. Main Coefficient Generator Routine (coefgen.c) ..... 46
A-3. Subroutine Find Ideal Index Vector (index.c) ..... 48
A-4. Geometry File — Declare All Global Constants (coefgen.h) ..... 49
A-5. Declare All Global Variables (coefgen.var) ..... 50
A-6. Matrix Pseudo Inverse Coefficients (initmat.c) ..... 51
A-7. Initialize All Variables (varinit.c) ..... 55
A-8. Find Least Squares Polynomial Fit (poly_fit.c) ..... 57
A-9. Matrix Inverter (inverse.c) ..... 59

# A-1. Main Program to Calculate Coefficients (main.c)

This is a listing of the code that calculates the coefficients needed by the fast focusing algorithm. It takes as inputs the geometry of the synthetic aperture radar (SAR) and image area, and it outputs the coefficient vectors required.

```
/*****************************************************************************

  Program : coefgen

  Description : This is the main  program to generate coefficients
                for fast focusing algorithm .


*****************************************************************************/

#include <malloc.h>
#include <stdio.h>
#include <math.h>
#include "coefgen.const"
#include "coefgen.var"

main()
{
 long cacheset;
 FILE *fp,*tstart_fp;
 char fn[80];
 long i;
 ia=(BOX_SIZE_AZIMUTH-4)/4;
 ib=(BOX_SIZE_RADIAL-3)/4;
/* initialize variables */
 coefgen_varinit();
/* initialize pseudo inver matrix */
 initmat();

 printf("Enter coefficient a  output file name ===> ");
 scanf("%s",fn);
 fp=fopen(fn,"w");
 if(fp==NULL)
  {
   printf("Error open output file \n");
   exit(1);
  }
/*
```

```
 printf("Enter data acquisition timing file name ===> ");
 scanf("%s",fn);
*/
/*
 tstart_fp=fopen("delaytime.dat","r");
 if(tstart_fp==NULL)
  {
   printf("Error open input file \n");
   exit(0);
  }
 read_delaytime(tstart_fp);
*/
 coefgen(fp);
}
```

## A-2. Main Coefficient Generator Routine (coefgen.c)

```
#include "coefgen.h"            /*geometry file*/
#include "coefgen.var"          /*list of variables*/
#include <stdio.h>
#include <math.h>
extern  index();
extern  mxmul_();
extern  poly_fit();
/* Note that Reference point is now taken at coordinate kpixel= n_azimuth/2-1  */
/*                                ipixel= n_radial/2 -1  */
/* At every position data were taken so that the reference point is at center  */
/* data buffer                                                                 */

void coefgen(fp)
FILE *fp;
{
 float input[NPOINT_APER/4];
 long i,j;
 long section,kbox,ibox,position,position_start,position_stop;
  long ipoint,kpoint;
 for(i=0;i<NPOINT_APER/n_section;i++)
  {
   input[i]=(float)i;
  }
 for(section=0;section<n_section;section++)
  {
   position_start=section*n_aper/n_section;
   position_stop=position_start+n_aper/n_section;
```

```
    for(kbox=0;kbox<NBOX_AZIMUTH;kbox++)
     {
      for(ibox=0;ibox<NBOX_RADIAL;ibox++)
       {
        printf("processing section %d ibox %d kbox %d\n",section,ibox,kbox);
        for(position=position_start;position<position_stop;position++)
         {
/******Calculating Actual index for all samples points in a box**********/
          i=0;
          for(ipoint=0;ipoint<NPOINT_BOXSAMPLE_RADIAL;ipoint++)
           {
            for(kpoint=0;kpoint<NPOINT_BOXSAMPLE_AZIMUTH;kpoint++)
             {
              ipixel=ibox*BOX_SIZE_RADIAL+ipoint*ib;
              jpixel=kbox*BOX_SIZE_AZIMUTH+kpoint*ia;

  d=((float)position-((float)n_aper/2.-1.))*(aper_length/((float)n_aper-1.)); /* distance from radar
to center of aperture */
  x=sqrt(rcenter_ref*rcenter_ref-radar_height2);
  r_ref=sqrt(radar_height2+x*x*c_theta_center_ref*c_theta_center_ref+(d-
x*s_theta_center_ref)*(d-x*s_theta_center_ref));
  rmin=r_ref-d_rcenter*((float)n_radial/2.-1.);
              index(&ipixel,&jpixel,&findex[i]);
              i++;
             }
           }
/* generate coef for this box at this position */
        mxmuls(mat,&mat_c_stride,&stride,findex,&findex_c_stride,&stride,coef,&coef_c_stride,
&stride,&n_coef,&i_one,&n_boxsample);

/* save in coefc[i][position]  */
          for(i=0;i<COEF_SIZE;i++)
           {
            coefc[i][position-position_start]=coef[i];
           }

/* for debug only */
/*   printf("pos= %d tstart= %.3e\n",position,2.0*rmin/c);  */
/* end debug */

         } /* position */
       for(i=0;i<COEF_SIZE;i++)
        {
         printf("Doing polyfit for coef %d\n",i);
poly_fit(input,&coefc[i][0],n_aper/n_section,deg[i],&coefa[i][0]);
```

```
            for(j=0;j<(deg[i]+1);j++)
              {
               fprintf(fp,"%e\n",coefa[i][j]);  /* save a[i] to file */
              }
           }
        } /* ibox */
       } /* kbox */
   } /* section */

} /* subroutine */
```

## A-3. Subroutine Find Ideal Index Vector (index.c)

```
/*************************************************************************/

 Subroutine:      index.c
 Input:
       ipixel          pixel radial coordinate
       jpixel          pixel angle coordinate
       a_span            angle spanned by the patch
       a_ofset          offset angle of the center line
       d               distance from middle position
       d_rcenter         sampling range
       dr8             (sampling range)/8
       rmincenter        min range rom center position
       hgt             height of building
       hgt2            square the height
       length           length of the building
 Output:
       findex           floating point index to ivalue of that pixel

*************************************************************************/
#include "coefgen.const"
#include "coefgen.var"
#include <math.h>

index(ipixel,kpixel,findex)
int *ipixel,*kpixel;
float *findex;
{

  theta=a_ofset-a_span/2.+(float)*kpixel*d_theta;
  c_theta=cos(theta);
  s_theta=sin(theta);
  rcenter=rmincenter+(float)*ipixel*d_rcenter;
```

```
  x=sqrt(rcenter*rcenter-radar_height2);
  r=sqrt(radar_height2+x*x*c_theta*c_theta+(d-x*s_theta)*(d-x*s_theta));
  *findex=(r-rmin)/dr8;
}
```

## A-4. Geometry File — Declare All Global Constants (coefgen.h)

```
#define NPOINT_AZIMUTH 600        /* number of bearing lines */
#define NPOINT_RADIAL  4095       /* number of radial lines  */
#define NPOINT_APER    2304       /* # of positions in aperture */
#define NPOINT_DATA    2048       /* number of original data points */
#define NPOINT_DATA_INTER NPOINT_DATA*8 /* number of data points after
                                   interpolated */
#define PI 3.141592654
#define RADAR_HEIGHT 60.          /* radar height in feet    */
#define A_OFSET      -15.0        /* offset angle from center line */
#define A_SPAN       54.0         /* spanning angle of the patch */
#define RMINCENTER   550.         /* range from center position to
                        nearest point on the patch */
#define RCENTER_REF  802.0        /* range (ft) of ref. point from center pos. */
#define THETA_CENTER_REF -15.0 /* angle (deg)of ref. point from center pos. */
#define SAMPLING_RATE 2.0e09      /* sampling frequency of signal  */
#define SAMPLING_PERIOD 5.0e-10   /* ts=1/fs                       */


#define BOX_SIZE_RADIAL    195
#define BOX_SIZE_AZIMUTH   100
#define NBOX_RADIAL    NPOINT_RADIAL/BOX_SIZE_RADIAL
#define NBOX_AZIMUTH NPOINT_AZIMUTH/BOX_SIZE_AZIMUTH

#define SECTION_SIZE 4            /* # of sections for one aperture */
#define COEF_SIZE   6             /* # of coeffs for curve fit  */
#define MAXDEG     3              /* maximum degree for poly. fit */
#define NPOINT_BOXSAMPLE_RADIAL 5   /* # of samples in a box in radial
                        direction              */
#define NPOINT_BOXSAMPLE_AZIMUTH 5  /* # of samples in azimuth direction
                        for every box          */
#define NPOINT_BOXSAMPLE
NPOINT_BOXSAMPLE_RADIAL*NPOINT_BOXSAMPLE_AZIMUTH
                    /* # of samples in a box for curve fit */
```

# A-5. Declare All Global Variables (coefgen.var)

```
struct complex { float r,i;};            /* define a complex type     */

/* Data Variables */
float pixel[NPOINT_AZIMUTH][NPOINT_RADIAL]; /* array of image          */
float data[NPOINT_DATA];                /* original data bufer       */
float data_inter[NPOINT_DATA_INTER];       /* interpolated data buffer  */
struct complex data_inter_fft[NPOINT_DATA_INTER/2]; /* Real->Complex Forward FFT
                                of data_inter          */
float filter_coef[NPOINT_DATA_INTER];      /* filter coefficients       */
struct complex filter_fft[NPOINT_DATA_INTER/2];    /* Real->Complex Forward FFT
                                of filter_coef         */
float nyquist_filter;        /* value of FFT of filter at Nyquist point  */
float nyquist_data;          /* value of FFT of data at Nyquist Point    */

/* Geometry Variables */
float a_ofset;             /* offset angle from the center line       */
float a_span;              /* spanning angle of the patch             */
float d_theta;             /* delta angle                             */
float x,theta;             /* coordinate of a pixel                   */
float c_theta,s_theta;       /* cosine and sine of theha              */
float d;                   /* distance from radar to center position
                           positive to the right, neg. to the left  */
float rcenter,rmincenter,rmaxcenter; /* range from center position      */
float rcenter_ref;           /* range from center pos. to ref point     */
float theta_center_ref;       /* angle of ref. point from center position */
float r_ref;               /* range from any pos. to ref. point       */
float d_rcenter;             /* sampling distance from center position  */
float dr8;                 /* (sampling distance)/8                   */
float r,rmin,rmax;           /* range from any position                 */
float aper_length;           /* length of aperture                      */
float radar_height;          /* height of radar                         */
float radar_height2;         /* square the height of radar              */

/* Radar Variables */
float c;                   /* speed of wave                           */
float ts;                  /* sampling period of signal               */
float fs;                  /* sampling frequency of signal            */

long n_azimuth;              /* # of points in azimuth direction        */
long n_radial;             /* # of points in radial direction         */
long n_aper;               /* # of positions in aperture              */
long pix_size;             /* # of points in pixel array              */
```

```
long n_data;                /* number of original data points          */
long n_data_inter;          /* number of interpolated data points      */
long n_data_inter_half;     /* 1/2 # of interpolated data points       */
long n_boxsample;           /* # of samples in a box for curve fit     */
long n_coef;                /* # of coefficients curve fit             */
long n_section;             /* # of sections for one aperture          */

/* SSL VAriables */
float f_zero;               /* floating point zero                     */
float f_one;                /* floating point 1                        */
long  i_one;                /* integer 1                               */
long stride;                /* stride for supercard ssl                */

/* working variables */
long n_position;
float e_index,a_index,error,maxerror,minerror;
long ierr_max,jerr_max,ierr_min,jerr_min;
long column_max,row_max,column_min,row_min;
long ia,ib;
long ipixel,jpixel;
float c_theta_center_ref,s_theta_center_ref;
float
mat[COEF_SIZE][NPOINT_BOXSAMPLE],findex[NPOINT_BOXSAMPLE],coef[COEF_SIZE];
float coefc[COEF_SIZE][NPOINT_APER/4]; /* c coef. for each section      */
float coefa[COEF_SIZE][MAXDEG+1];      /* a coef. for each c            */
long deg[COEF_SIZE];                   /* degree for each coefc         */
long mat_c_stride,findex_c_stride,coef_c_stride;
float tstart[NPOINT_APER];    /* time from trigger to start data acq. at */
                              /*     each position                       */
```

## A-6. Matrix Pseudo Inverse Coefficients (initmat.c)

```
#include <stdio.h>
#include <math.h>
#include "coefgen.const"
#include "coefgen.var"

initmat()
{
int i,j;
float a,b;

a=((float)BOX_SIZE_AZIMUTH-4.)/4.;
b=((float)BOX_SIZE_RADIAL-3.)/4.;
```

## Appendix A

```
mat[0][0]=93./175.;
mat[0][1]=27./175.;
mat[0][2]=(-9.)/175.;
mat[0][3]=(-3.)/35.;
mat[0][4]=9./175.;
mat[0][5]=62./175.;
mat[0][6]=18./175.;
mat[0][7]=(-6.)/175.;
mat[0][8]=(-2.)/35.;
mat[0][9]=6./175.;
mat[0][10]=31./175.;
mat[0][11]=9./175.;
mat[0][12]=(-3.)/175.;
mat[0][13]=(-1.)/35.;
mat[0][14]=3./175.;
mat[0][15]=0.;
mat[0][16]=0.;
mat[0][17]=0.;
mat[0][18]=0.;
mat[0][19]=0.;
mat[0][20]=(-31.)/175.;
mat[0][21]=(-9.)/175.;
mat[0][22]=3./175.;
mat[0][23]=1./35.;
mat[0][24]=(-3.)/175.;

mat[1][0]=(-81.)/(175.*a);
mat[1][1]=39./(350.*a);
mat[1][2]=12./(35.*a);
mat[1][3]=81./(350.*a);
mat[1][4]=(-39.)/(175.*a);
mat[1][5]=(-54.)/(175.*a);
mat[1][6]=13./(175.*a);
mat[1][7]=8./(35.*a);
mat[1][8]=27./(175.*a);
mat[1][9]=(-26.)/(175.*a);
mat[1][10]=(-27.)/(175.*a);
mat[1][11]=13./(350.*a);
mat[1][12]=4./(35.*a);
mat[1][13]=27./(350.*a);
mat[1][14]=(-13.)/(175.*a);
mat[1][15]=0.;
mat[1][16]=0.;
mat[1][17]=0.;
mat[1][18]=0.;
```

```
mat[1][19]=0.;
mat[1][20]=27./(175.*a);
mat[1][21]=(-13.)/(350.*a);
mat[1][22]=(-4.)/(35.*a);
mat[1][23]=(-27.)/(350.*a);
mat[1][24]=13./(175.*a);

mat[2][0]=3./(35.*pow(a,2.));
mat[2][1]=(-3.)/(70.*pow(a,2.));
mat[2][2]=(-3.)/(35.*pow(a,2.));
mat[2][3]=(-3.)/(70.*pow(a,2.));
mat[2][4]=3./(35.*pow(a,2.));
mat[2][5]=2./(35.*pow(a,2.));
mat[2][6]=(-1.)/(35.*pow(a,2.));
mat[2][7]=(-2.)/(35.*pow(a,2.));
mat[2][8]=(-1.)/(35.*pow(a,2.));
mat[2][9]=2./(35.*pow(a,2.));
mat[2][10]=1./(35.*pow(a,2.));
mat[2][11]=(-1.)/(70.*pow(a,2.));
mat[2][12]=(-1.)/(35.*pow(a,2.));
mat[2][13]=(-1.)/(70.*pow(a,2.));
mat[2][14]=1./(35.*pow(a,2.));
mat[2][15]=0.;
mat[2][16]=0.;
mat[2][17]=0.;
mat[2][18]=0.;
mat[2][19]=0.;
mat[2][20]=(-1.)/(35.*pow(a,2.));
mat[2][21]=1./(70.*pow(a,2.));
mat[2][22]=1./(35.*pow(a,2.));
mat[2][23]=1./(70.*pow(a,2.));
mat[2][24]=(-1.)/(35.*pow(a,2.));

mat[3][0]=(-31.)/(175.*b);
mat[3][1]=(-9.)/(175.*b);
mat[3][2]=3./(175.*b);
mat[3][3]=1./(35.*b);
mat[3][4]=(-3.)/(175.*b);
mat[3][5]=(-31.)/(350.*b);
mat[3][6]=(-9.)/(350.*b);
mat[3][7]=3./(350.*b);
mat[3][8]=1./(70.*b);
mat[3][9]=(-3.)/(350.*b);
mat[3][10]=0.;
mat[3][11]=0.;
```

## Appendix A

```
mat[3][12]=0.;
mat[3][13]=0.;
mat[3][14]=0.;
mat[3][15]=31./(350.*b);
mat[3][16]=9./(350.*b);
mat[3][17]=(-3.)/(350.*b);
mat[3][18]=(-1.)/(70.*b);
mat[3][19]=3./(350.*b);
mat[3][20]=31./(175.*b);
mat[3][21]=9./(175.*b);
mat[3][22]=(-3.)/(175.*b);
mat[3][23]=(-1.)/(35.*b);
mat[3][24]=3./(175.*b);

mat[4][0]=27./(175.*a*b);
mat[4][1]=(-13.)/(350.*a*b);
mat[4][2]=(-4.)/(35.*a*b);
mat[4][3]=(-27.)/(350.*a*b);
mat[4][4]=13./(175.*a*b);
mat[4][5]=27./(350.*a*b);
mat[4][6]=(-13.)/(700.*a*b);
mat[4][7]=(-2.)/(35.*a*b);
mat[4][8]=(-27.)/(700.*a*b);
mat[4][9]=13./(350.*a*b);
mat[4][10]=0.;
mat[4][11]=0.;
mat[4][12]=0.;
mat[4][13]=0.;
mat[4][14]=0.;
mat[4][15]=(-27.)/(350.*a*b);
mat[4][16]=13./(700.*a*b);
mat[4][17]=2./(35.*a*b);
mat[4][18]=27./(700.*a*b);
mat[4][19]=(-13.)/(350.*a*b);
mat[4][20]=(-27.)/(175.*a*b);
mat[4][21]=13./(350.*a*b);
mat[4][22]=4./(35.*a*b);
mat[4][23]=27./(350.*a*b);
mat[4][24]=(-13.)/(175.*a*b);

mat[5][0]=(-1.)/(35.*pow(a,2.)*b);
mat[5][1]=1./(70.*pow(a,2.)*b);
mat[5][2]=1./(35.*pow(a,2.)*b);
mat[5][3]=1./(70.*pow(a,2.)*b);
mat[5][4]=(-1.)/(35.*pow(a,2.)*b);
```

```
mat[5][5]=(-1.)/(70.*pow(a,2.)*b);
mat[5][6]=1./(140.*pow(a,2.)*b);
mat[5][7]=1./(70.*pow(a,2.)*b);
mat[5][8]=1./(140.*pow(a,2.)*b);
mat[5][9]=(-1.)/(70.*pow(a,2.)*b);
mat[5][10]=0.;
mat[5][11]=0.;
mat[5][12]=0.;
mat[5][13]=0.;
mat[5][14]=0.;
mat[5][15]=1./(70.*pow(a,2.)*b);
mat[5][16]=(-1.)/(140.*pow(a,2.)*b);
mat[5][17]=(-1.)/(70.*pow(a,2.)*b);
mat[5][18]=(-1.)/(140.*pow(a,2.)*b);
mat[5][19]=1./(70.*pow(a,2.)*b);
mat[5][20]=1./(35.*pow(a,2.)*b);
mat[5][21]=(-1.)/(70.*pow(a,2.)*b);
mat[5][22]=(-1.)/(35.*pow(a,2.)*b);
mat[5][23]=(-1.)/(70.*pow(a,2.)*b);
mat[5][24]=1./(35.*pow(a,2.)*b);

}
```

## A-7. Initialize All Variables (varinit.c)

```
/*****************************************************************************

 Subroutine: coefgen_varinit

 Description : initialize all neccessary variables

*****************************************************************************/

#include <stdio.h>
#include <math.h>
#include "coefgen.const"
#include "coefgen.var"
coefgen_varinit()
{
 n_azimuth=NPOINT_AZIMUTH;            /* # of points in azi. direction */
 n_radial=NPOINT_RADIAL;              /* # of points in rad. direction */
 pix_size=n_azimuth*n_radial; /* Pixel array size            */
 n_data=NPOINT_DATA;                  /* # of orig. data points      */
 n_data_inter=NPOINT_DATA_INTER;      /* # of interpolated. data points*/
 n_data_inter_half=NPOINT_DATA_INTER/2; /* 1/2 # of inter. data points   */
```

## Appendix A

```
 n_aper=NPOINT_APER;                /* # of points in the aperture   */
 n_boxsample=NPOINT_BOXSAMPLE;        /* # of samples in box for curve fit */
 n_coef=COEF_SIZE;                 /* # of coeffs for curve fit     */
 n_section=SECTION_SIZE;            /* # of sections for one aperture*/

/* Radar Variables */
 fs=SAMPLING_RATE;                  /* Sampling Frequency          */
 ts=SAMPLING_PERIOD;                /* Sampling period             */
 c=3.e8;                        /* Speed of wave               */

/* Geometry variables  */
 a_ofset=A_OFSET*PI/180.;           /* Offset angle from center line */
 a_span=A_SPAN*PI/180.;             /* Spanning angle of the patch   */
 d_theta=a_span/((float)n_azimuth-1.); /* delta theta         */
 aper_length=(n_aper-1)*2.0*.0254;    /* length of aperture          */
 radar_height=RADAR_HEIGHT*12.*0.0254;  /* height of radar            */
 radar_height2=radar_height*radar_height; /* square the height */
 rmincenter=RMINCENTER*12.*0.0254;     /* min range from center position
                                 to reference point          */
 d_rcenter=ts*c/(2.0*2.0);         /* sampling range              */
                                  /* project back to 2*n_data samples */
 dr8=ts*c/(2.0*8.0);         /* interpolated 16K buffer */

/*
 theta_center=a_ofset-a_span/2.+((float)n_azimuth/2.-1.)*d_theta;
 c_theta_center=cos(theta_center);
 s_theta_center=sin(theta_center);
 rcenter_ref=rmincenter+d_rcenter*((float)n_radial/2.-1.);
*/
 rcenter_ref=RCENTER_REF*12.*.0254;  /* range of ref. point from center pos. */
 theta_center_ref=THETA_CENTER_REF*PI/180.; /* angle of ref. point from center
                                 position                    */
 c_theta_center_ref=cos(theta_center_ref);
 s_theta_center_ref=sin(theta_center_ref);

/* SSL Variables  */
 f_zero=0.;                     /* floating point zero         */
 stride=1;                      /* stride for ssl              */
 f_one=1.0;                      /* floating point 1            */
 i_one=1;                       /* integer 1                   */
 mat_c_stride=n_boxsample;           /* column stride for mat        */
 findex_c_stride=1;                 /* column stride for findex      */
 coef_c_stride=1;                  /* column stride for coeff       */
 deg[0]=3;                      /* degree for first coefc       */
 deg[1]=3;
```

```
 deg[2]=3;
 deg[3]=3;
 deg[4]=2;
 deg[5]=2;

/* xgints_(pixel,&f_zero,&pix_size,&stride); */ /* clear pixel[][] */

}
```

## A-8. Find Least Squares Polynomial Fit (poly_fit.c)

```
/*************************************************************************
   Subroutine: poly_fit.c
   Description:
            Perform data fit to polynomial
   Input:    x[n]        input array
            y[n]        input array
            n           number of elements
            deg         degree of poly.
   output    coeff[deg+1]     coefficients of poly
*************************************************************************/
#include <stdio.h>
#include <malloc.h>
#include <math.h>
void poly_fit(x,y,n,deg,coeff)
float x[],y[],coeff[];
long n,deg;
{
 double *dx,*dy,*dcoeff;
 double *dX,*dXtrans,*dA,*dB,*dC;
 long m,i,j,one;
 one=1;
 m=deg+1;
 dx=malloc(n*sizeof(double));
 if(dx==NULL)
  {
  printf("Memory allocation Error !!!\n");
  exit(1);
  }
 dy=malloc(n*sizeof(double));
 if(dy==NULL)
  {
  printf("Memory allocation Error !!!\n");
  exit(1);
  }
```

## Appendix A

```
 dcoeff=malloc(m*sizeof(double));
 if(dcoeff==NULL)
  {
   printf("Memory allocation Error !!!\n");
   exit(1);
  }
 dX=malloc(n*m*sizeof(double));
 if(dX==NULL)
  {
   printf("Memory allocation Error !!!\n");
   exit(1);
  }
 dXtrans=malloc(n*m*sizeof(double));
 if(dX==NULL)
  {
   printf("Memory allocation Error !!!\n");
   exit(1);
  }
 dA=malloc(m*m*sizeof(double));
 if(dA==NULL)
  {
   printf("Memory allocation Error !!!\n");
   exit(1);
  }
 dB=malloc(m*m*sizeof(double));
 if(dB==NULL)
  {
   printf("Memory allocation Error !!!\n");
   exit(1);
  }
 dC=malloc(m*sizeof(double));
 if(dx==NULL)
  {
   printf("Memory allocation Error !!!\n");
   exit(1);
  }

 for(i=0;i<n;i++)
  {
   dx[i]=(double)x[i];
   dy[i]=(double)y[i];
  }
 for(i=0;i<n;i++)
  {
   dX[i*m]=1.0;
```

```
  }
 for(j=1;j<m;j++)
  {
   for(i=0;i<n;i++)
    {
     dX[j+i*m]=pow(dx[i],(double)j);
    }
  }
 mxtrans(dX,n,m,dXtrans);
 mxmuld(dXtrans,&n,&one,dX,&m,&one,dA,&m,&one,&m,&m,&n);
 mat_inverse(dA,dB,m);
 mxmuld(dXtrans,&n,&one,dy,&one,&one,dC,&one,&one,&m,&one,&n);
 mxmuld(dB,&m,&one,dC,&one,&one,dcoeff,&one,&one,&m,&one,&m);
 for(i=0;i<m;i++)
  {
   coeff[i]=(float)dcoeff[i];
  }
 free(dx);
 free(dy);
 free(dA);
 free(dB);
 free(dC);
 free(dX);
 free(dXtrans);
 free(dcoeff);
}
```

## A-9. Matrix Inverter (inverse.c)

```
#include <stdio.h>

mat_inverse(source_mat,dest_mat,size_mat)
double source_mat[];
double dest_mat[];
long size_mat;
{
 long n,i,j,k,ki,count;
 double b,b1;
 double err=0.0001;
 double a[100][100];
 n=size_mat;
 count=0;
```

## Appendix A

```
for(i=0;i<size_mat;i++)
  {
   for(j=0;j<size_mat;j++)
     {
      a[i+1][j+1]=source_mat[count++];
     }
  }

for(i=1;i<=n;i++)
  {
    for(j=n+1;j<=2*n;j++)
       {
        if((j-n-i)==0)
           { a[i][j]=1.0; }
        else
           { a[i][j]=0.;  }
       }
  }

for(k=1;k<=n-1;k++)

  {
   b=a[k][k];
   ki=k;
   for(i=k+1;i<=n;i++)
     {
      if((abs(b)-abs(a[i][k]))<0)
        {
         b=a[i][k];
         ki=i;
        }
     }

   if((abs(b)-err)<0)
     {
      printf("Error matrix inverse , matrix is singular\n");
      exit(1);
     }

   if((ki-k)!=0)
     {
      for(j=k;j<=2*n;j++)
        {
```

```
            b1=a[k][j];
            a[k][j]=a[ki][j];
            a[ki][j]=b1;
          }
        }

      for(j=k+1;j<=2*n;j++)
       {
        a[k][j]=a[k][j]/b;
       }

      for(i=k+1;i<=n;i++)
       {
        for(j=k+1;j<=2*n;j++)
         {
          a[i][j]=a[i][j]-a[i][k]*a[k][j];
         }
       }
     }
   for(j=n+1;j<=2*n;j++)
    {
     a[n][j]=a[n][j]/a[n][n];
    }

   for(k=n-1;k>=1;k=k-1)
    {
     for(j=n+1;j<=2*n;j++)
      {
       for(i=k+1;i<=n;i++)
        {
         a[k][j]=a[k][j]-a[k][i]*a[i][j];
        }
      }
    }

  count=0;
  for(i=0;i<size_mat;i++)
   {
    for(j=0;j<size_mat;j++)
     {
      dest_mat[count++]=a[i+1][size_mat+j+1];
     }
   }
}
```

# Appendix B. UWB SAR Focusing Program

## Appendix B

### Contents

page


**B-1. Main Program for SUN Computer (focus.c)** ..... 65
**B-2. Main Program for Multiple Array Processors (sc.c)** ..... 70
**B-3. Routines Used to Interpolate (inter.c)** ..... 73
**B-4. Initialize Interpolation Filter (filtinit.c)** ..... 77
**B-5. Main Back Projection Focusing Routing (bp.c)** ..... 78
**B-6. Declare All Global Constants (Focus.h)** ..... 82
**B-7. Declare All Global Variables (Focus.var)** ..... 83
**B-8. Initialize All Global Variables on Both Processors** ..... 85

# B-1. Main Program for SUN Computer (focus.c)

This is a listing of the code that focuses the ultra-wideband (UWB) synthetic aperture radar (SAR) data. It takes as inputs the coefficient vectors generated by the code listed in appendix A and the SAR data, and it outputs a focused image.

```
/*********************************************************************

 program: focus.c
 description:
       this programs reads radar data from file, interpolates
       data using convolution in freq. domain, and projects
       data back to the polar grid on the ground. The indeces
       used in back projection are computed from the file
       which contains coefficients for a particular geometry

*********************************************************************/

#include <stdio.h>
#include "focus.h"
#include "focus.var"

 float data[NPOINT_DATA]; /* data buffer for host */
 struct cstype *cs[NO_SUPERCARD];

main ()
 {
  float tstart_err[NPOINT_APER]; /* tstart-tstart_trunc for each position */
  FILE *inputfile;   /* input file contains radar data          */
  FILE *outputfile;  /* output file contains focused image       */
  FILE *filterfile;  /* input file contains filter coefficients  */
  FILE *coeffile;    /* input file contains back projection      */
                /*                coefficients        */
  FILE *tstart_err_file; /* input file contains tstart-tstart_trunc  */
  FILE *junkfile;
  char inputname[80];
  char outputname[80];
  char filtername[80];
  char coefname[80];
  char tstart_err_name[80];
  char temp_string[20];
  long section,position,box,coefc_order,coefa_order;
  long i,j;        /* just working variable     */
```

## Appendix B

```
  float f_position;
  long data_ready=1;  /* maibox to indicate data ready  host ---> SC */
  long data_consumed=2; /* mailbox to indicate data consumed  host <--- SC */
  long msg;          /* message read from mailbox  */
  long one=1;        /* nonzero message to put in mailbox */
  long cacheset=1;   /* cacheset=0 turn off cache  */
  long numpar=1;
  float pixel_zero[BOX_SIZE_AZIMUTH][NPOINT_RADIAL];


/*** open supercards ********************************************/
  for (i=0;i<NO_SUPERCARD;i++)
   {
    cs[i]=(struct cstype *)xlubgn_(0,&cacheset,"sc.lo");
   }

/*** initialize supercard variables ****************************/
  focus_varinit();

/*** initialize supercards' id **********************************/
  for(i=0;i<NO_SUPERCARD;i++)
   {
    cs[i]->supercard_id=i;
   }

/*** Enter Input ************************************************/
 printf("Enter input file name ( .raw )===> ");
 scanf("%s",inputname);
 printf("Enter output file name ( .focus)===> ");
 scanf("%s",outputname);
/*
 printf("Enter back projection coefficient file name (coefa.dat)===> ");
 scanf("%s",coefname);
 printf("Enter start time difference file name (delaytime_diff.dat)===> ");
 scanf("%s",tstart_err_name);
*/
 strcpy(coefname,"coefa.dat");
 strcpy(tstart_err_name,"delaytime_diff.dat");

 printf("Do you want hamming weight accross the aperture (y or n) ===> ");
 scanf("%s",temp_string);

 for(j=0;j<NO_SUPERCARD;j++)
  {
   cs[j]->ham_flag=0;
```

```
  if(temp_string[0]=='y')
    {
     cs[j]->ham_flag=1;
    }
 }

 strcpy(filtername,"filter.dat");
 if((inputfile=fopen(inputname,"rb"))==NULL)
   {
    printf("Error open input file\n");
    exit(1);
   }
 if((outputfile=fopen(outputname,"wb"))==NULL)
   {
    printf("Error open output file\n");
    exit(1);
   }
 if((filterfile=fopen(filtername,"r"))==NULL)
   {
    printf("Error open filter file\n");
    exit(1);
   }
 if((coeffile=fopen(coefname,"r"))==NULL)
   {
    printf("Error open back projection coefficient file\n");
    exit(1);
   }
 if((tstart_err_file=fopen(tstart_err_name,"r"))==NULL)
   {
    printf("Error open tstart difference file\n");
    exit(1);
   }

/*** Read filter coefficients ********************************/
/*** into 1st supercard and copy to others supercards **********/
 printf(">>> Reading filter coefficients\n");
 for(i=0;i<FILTER_LENGTH;i++)
  {
   fscanf(filterfile,"%f",&cs[0]->filter_coef[i]);
  }

 for(j=1;j<NO_SUPERCARD;j++)
  {
   for(i=0;i<FILTER_LENGTH;i++)
    {
```

## Appendix B

```
      cs[j]->filter_coef[i]=cs[0]->filter_coef[i];
      }
   }


/*** Read in tstart_err for all positions to first supercard *****/
/*** and then copy to other supercards ***************************/
 printf(">>> Reading tstart_err data \n");
 for(i=0;i<NPOINT_APER;i++)
  {
   fscanf(tstart_err_file,"%d %f",&j,&(cs[0]->tstart_err[i]));
  }
 for(j=1;j<NO_SUPERCARD;j++)
  {
   for(i=0;i<NPOINT_APER;i++)
    {
     cs[j]->tstart_err[i]=cs[0]->tstart_err[i];
    }
  }


/*** Read in coefficients a for back projection ******************/
/*** to the first supercard and then copy to other supercards ****/
 printf(">>> Reading backprojection coefficients \n");
 for(section=0;section<SECTION_SIZE;section++)
  {
   for(box=0;box<NBOX_RADIAL*NBOX_AZIMUTH;box++)
    {
     for(coefc_order=0;coefc_order<COEF_SIZE;coefc_order++)
      {
       for(coefa_order=0;coefa_order<(cs[0]->deg[coefc_order]+1);coefa_order++)
       {
fscanf(coeffile,"%f",&(cs[0]->coefa[section][box][coefc_order][coefa_order]));
        for(j=1;j<NO_SUPERCARD;j++)
          {
           cs[j]->coefa[section][box][coefc_order][coefa_order]=
              cs[0]->coefa[section][box][coefc_order][coefa_order];
          }
       } /* coefa_order */
      }  /* coefc_order */
    }    /* box        */
  }      /* section    */

/*** Start supercards ********************************************/
```

```
 printf(">>> Starting supercard programs \n");
 for(j=0;j<NO_SUPERCARD;j++)
  {
   xrcall_("sc",&numpar,&cs[j]->dummy);
  }

/*** For every position, the host computer reads data for that ***/
/*** position and copy data into each supercard memory. **********/
/*** There are two mail boxes used betwwen host and each *********/
/*** supercard to provide synchronization ************************/

 for(section=0;section<SECTION_SIZE;section++)
  {
   for(position=0;position<NPOINT_APER/SECTION_SIZE;position++)
    {
     printf(">>>>>processing section %d pos %d <<<<<<<\n",section,position);

/*** Read in data for a position *********************************/
     fread(data,sizeof(data),1,inputfile);

     for(j=0;j<NO_SUPERCARD;j++)
       {
/*** Copy data in each supercard memory **************************/
       for(i=0;i<NPOINT_DATA;i++)
         {
          cs[j]->data_temp[i]=data[i];
         }
/*** Send mail to supercard to inform data is ready **************/
       xlnwxmt_(cs[j],&data_ready,&one);
       }
/*** Wait until data is consumed by supercards *******************/
     for(j=0;j<NO_SUPERCARD;j++)
       {
       xlwtrec_(cs[j],&data_consumed,&msg);
       }
    }    /* position */
  }      /* section  */


/*** Check for supercards to finish all processing ******************/
 for(j=0;j<NO_SUPERCARD;j++)
  {
   xldone_(cs[j]);
  }
```

## Appendix B

```
/*** Save resulting radar image ************************************/
 for(j=0;j<BOX_BASE_START;j++)
  {
   printf(">>>Prepad zero to output file \n");
   fwrite(pixel_zero,sizeof(pixel_zero),1,outputfile);
  }
 printf(">>>Save image to output file \n");
 for(j=0;j<NO_SUPERCARD;j++)
  {
   fwrite(cs[j]->pixel,
    sizeof(float)*BOX_SIZE_AZIMUTH*NPOINT_RADIAL*NO_KBOX_PROCESS/
NO_SUPERCARD,
    1,outputfile);
  }
 for(j=BOX_BASE_START+NO_KBOX_PROCESS;j<NBOX_AZIMUTH;j++)
  {
   printf(">>>Postpad zero to output file \n");
   fwrite(pixel_zero,sizeof(pixel_zero),1,outputfile);
  }

 fclose(inputfile);
 fclose(outputfile);

/*** Closing all supercards ****************************************/
 for(j=NO_SUPERCARD-1;j>=0;j--)
  {
   xlclos_(cs[j]);
  }

} /* end main */
```

# B-2. Main Program for Multiple Array Processors (sc.c)

```
/*****************************************************************

 Program: sc.c

 Description: This is the main program to be executed by each of
   the CSPI i860 array processor.

*****************************************************************/
#include "focus.h"
#include "focus.var"
struct cstype *cs;
```

```
void sc(dummy)
long *dummy;
 {
  void focus_varinit();
  void filter_init();
  void haminit();
  void hamwt();
  void erase();
  void inter();
  void fix_data_pointer();
  void poly();
  void bp();
  void xwtrec_();
  void xnwxmt_();
  void xvmov_();
  void xvclr_();

  float *data_pointer;    /* pointer to actual data              */
  long section,position,box,coefc_order,kbox,ibox;
  long data_ready=1; /* maibox to indicate data ready  host ---> SC */
  long data_consumed=2; /* mailbox to indicate data consumed  host <--- SC */
  long msg;          /* message read from mailbox  */
  long one=1;        /* nonzero message to put in mailbox */
  float f_position;
  long boxbase;

/*** initialize supercard pointer ***********************************/
  cs=0;

/*** initialize image with zeros ************************************/
  xvclr_(cs->pixel,&cs->pix_size,&cs->i_one);

/*** initialize filter **********************************************/
  filter_init();

/*** initialize hamming weight coefficients *************************/
  haminit(cs->ham_coef,NPOINT_APER);

/*** Start forming image ********************************************/

boxbase=BOX_BASE_START+cs->supercard_id*NO_KBOX_PROCESS/NO_SUPERCARD;

for(section=0;section<SECTION_SIZE;section++)
 {
  for(position=0;position<NPOINT_APER/SECTION_SIZE;position++)
   {
```

## Appendix B

```
/*** Wait for new data from host computer **************************/
   xwtrec_(&data_ready,&msg);

/*** Copy data to working buffer ***********************************/
   xvmov_(cs->data,cs->data_temp,&cs->n_data,&cs->i_one,&cs->i_one);

/*** Signal host computer that data are consumed *******************/
   xnwxmt_(&data_consumed,&one);

   f_position=(float)position;

/*** Hamming weight data if needed *********************************/
   if(cs->ham_flag==1)
    {
     hamwt(cs->data,cs->n_data,section*NPOINT_APER/SECTION_SIZE+position,
          cs->ham_coef);
    }

/*** zeros last portion of data for circular convolution ***********/
   erase(cs->data,cs->n_data,32);

/*** Interpolate data **********************************************/
   inter(cs->data,cs->n_data,cs->data_inter,
         cs->n_data_inter,cs->filter_fft);

   fix_data_pointer(&data_pointer,&cs->data_inter[0],
        cs->tstart_err[NPOINT_APER/SECTION_SIZE*section+position],
         cs->ts,8);

   for(kbox=boxbase;kbox<(boxbase+NO_KBOX_PROCESS/NO_SUPERCARD);kbox++)
    {
    for(ibox=IBOX_START;ibox<(IBOX_START+NO_IBOX_PROCESS);ibox++)
     {
     box=kbox*NBOX_RADIAL+ibox;

/*** Generate coefficients for back projection *******************/
      for(coefc_order=0;coefc_order<COEF_SIZE;coefc_order++)
       {
        poly(&f_position,&cs->coefc[coefc_order],1,
           &cs->coefa[section][box][coefc_order][0],
           cs->deg[coefc_order]);
      } /* coefc_order */

/*** Perform back projection ************************************/
      bp(cs->pixel,data_pointer,cs->n_data_inter,cs->coefc,
```

```
        boxbase,kbox,ibox,
        cs->k_pix_size,cs->i_pix_size,cs->i_box_size);

      } /* i_box */
    }  /* k_box */
  }    /* position */
 }     /* section  */

} /* end supercard program */
```

## B-3. Routines Used to Interpolate (inter.c)

```
#include <math.h>
#include "focus.h"
#include "focus.var"
/*****************************************************************************
 Subroutine: haminit.c
 Description:
        this subroutine initialize array of hamming weighting coeffs
        accross the aperture.
 Input:    float ham_coef[NPOINT_APER]
        long  npoint     Number of points for hamming coeffs
 Output:   float ham_coef[NPOINT_APER] is initialized
*****************************************************************************/
haminit(ham_coef,npoint)
float *ham_coef;
long npoint;
{
 long i;
 for(i=0;i<npoint;i++)
   {
    ham_coef[i]=.54-.46*cos(2.*3.1415927*(float)i/(float)npoint);
   }
}


/*****************************************************************************
 Subroutine: hamwt.c

 Description:
        this subroutine takes the radar signal at a position and
        multiplies the entire signal array with the hamming coef at
        that position.
```

## Appendix B

```
 Input:
        float data[NPOINT_DATA]
        long  npoint_data
        long  position
        float ham_coef[];

 Output:
        float data[NPOINT_DATA]

*************************************************************************/

hamwt(data,npoint_data,position,ham_coef)
float *data,*ham_coef;
long npoint_data,position;
{
 long i;
 for(i=0;i<npoint_data;i++)
   {
    data[i]=data[i]*ham_coef[position];
   }
}



/*************************************************************************
 Subroutine: inter.c
 Description:
        This routine performs FIR interpolation by using convolution
        in the frequency domain. The input buffer contains 2K of data
        and is interpolated into 16K of data. The FIR filter has
        breakpoints at .95 Ghz and 1.05 Ghz
                # of taps = 255
                name of filter coeff. file : filter.dat
 Input:    data[NPOINT_DATA]
        complex filter_fft[NPOINT_DATA_INTER/2]
        float   nyquist_filter
 Output:    data_inter[NPOINT_DATA_INTER]
*************************************************************************/

inter()
{
 void xvclr_();
 void xfrf_();
 void xcvmls_();
 void xfri_();
```

```
 extern struct cstype *cs;
 int i;

/* clear data_inter buffer */
 xvclr_(cs->data_inter,&cs->n_data_inter,&cs->i_one);

/* interleave data into data-inter buffer */
 for(i=0;i<NPOINT_DATA;i++)
  {
   cs->data_inter[i*8]=cs->data[i];
  }

/* FFT of interleaved data */
 xfrf_(cs->data_inter_fft,&cs->nyquist_data,cs->data_inter,
      &cs->n_data_inter_half);

/* FFT of interpolated data (multiply with filter in freq. domain)*/
 xcvmls_(cs->data_inter_fft,&cs->f_one,cs->data_inter_fft,
       cs->filter_fft,&cs->n_data_inter_half);
 cs->nyquist_data=cs->nyquist_data*cs->nyquist_filter;

/* inverse FFT to get interpolated data */
 xfri_(cs->data_inter,&cs->nyquist_data,cs->data_inter_fft,
      cs->data_inter_fft,&cs->n_data_inter_half);

}


/*************************************************************************

  Subroutine: fix_data_pointer

  Description: This subroutine fixes the pointer to the start of
           data_inter buffer. The pointer needed to be adjusted
           due to the following reasons:

           (a) The linear FIR interpolation filter has phase
              shift and the actual data point starts at bin 127
              of the data_inter buffer ( FIR has 255 coeffs )
           (b) The actual time to start data acquisition has to
              be rounded of to 1 nsec resolution for the scope
              DSA602. However the backprojection algorithm uses
              the exact time since the indeces have to be smooth
              for curve fit.
```

## Appendix B

```
  Input:
        float  *data_inter    ( address of data_inter buffer )
        float  tstart_err    ( tstart-tstart_trunc        )
        float  ts          ( radar sampling period       )
        long   inter_factor  (interpolation factor        )
  Output:
        float  *data_pointer ( actual start pointer for data )

*************************************************************************/

fix_data_pointer(pointer,data_inter,tstart_err,ts,inter_factor)
float **pointer;
float *data_inter;
float tstart_err;
float ts;
long inter_factor;
 {
  long i;
  double float1,float2;
  unsigned long index_ofset;
  float1=modf(tstart_err*inter_factor/ts,&float2);
  if(float1>=.5) float2=float2+1.;
  index_ofset=(unsigned long)127+(unsigned long)float2;
  *pointer=data_inter+index_ofset;

/* zeros fill first 128 points of data_inter since phase shift from linear filter   */
  for(i=0;i<128;i++)
   {
    data_inter[i]=0.;
   }

/* zeros fill the last 128 points of data_inter plus 8 points for tstart_err      */
  for(i=0;i<136;i++)
   {
    data_inter[NPOINT_DATA_INTER+i]=0.;
   }

/* first point and last point of actual data array are zeros for backprojection */
  **pointer=0.;
  *(*pointer+NPOINT_DATA_INTER-1)=0.;
 }


/*************************************************************************
```

Subroutine: erase

Description: Zero out the last nzero points of input buffer

```
*****************************************************************************/

erase(buffer,ndata,nzero)
float *buffer;
long ndata,nzero;
{
 long i;
 for(i=ndata-nzero;i<ndata;i++)
  {
   buffer[i]=0.;
  }
}
```

## B-4. Initialize Interpolation Filter (filtinit.c)

```
/*****************************************************************************

 This subroutine performs the following :
      1- Read filter coefficients from file to filter buffer
      2- Zerro pad the filter buffer to the size of interpolated data
         buffer size
      3- Perform Real Forward FFT of filter buffer to prepare for
         interpolation

/*****************************************************************************/
#include <stdio.h>
#include "focus.h"
#include "focus.var"

extern struct cstype *cs;

void filter_init()
{
 void xfrf_();
 long i;

/* zero pad filter coeff                    */
 for(i=FILTER_LENGTH;i<NPOINT_DATA_INTER;i++)
  {
   cs->filter_coef[i]=0.;
```

```
  }

/* take fft of filter coef for each supercard */
 for(i=0;i<NO_SUPERCARD;i++)
  {
   xfrf_(cs->filter_fft,&cs->nyquist_filter,
       cs->filter_coef,&cs->n_data_inter_half);
  }

}
```

## B-5. Main Back Projection Focusing Routing (bp.c)

```
#include <math.h>
/*****************************************************************************
  Subroutine: poly.c
  DEscription:
          Calculates values of polynomials
  Input:    x[]        input array
          n          number of elements
          coeff[]     coefficients of poly
          deg         degree of poly
  Output:   y[]        output array
*****************************************************************************/

poly(x,y,n,coeff,deg)
 float *x,*y,*coeff;
 long n,deg;
 {
  long i,j;
  for(i=0;i<n;i++)
   {
   y[i]=coeff[0];
   for(j=1;j<(deg+1);j++)
    {
    y[i]=y[i]+coeff[j]*pow(x[i],(float)j);
    }
   }
 }
/*****************************************************************************

    Subroutine: poly2.c
    Description: calculate y=c0+c1*x+c2*x^2
              where x=[0,1,2,...,n-1]
```

```
    Input:
        float c[3]      coefficients
        long n (max : 1000)  number of points
    Output:
       float y[]       output vector
*****************************************************************************/

#define MAX_ELEMENT  1000

void poly2(y,n,c)
float y[],c[];
long n;
{
 void xvrmp_();
 void xdintg_();
 float temp;
 float y1[MAX_ELEMENT];
 float y1_init,y1_inc;
 y1_init=c[1]-c[2];
 y1_inc=2.*c[2];
 xvrmp_(y1,&temp,&y1_init,&y1_inc,&n);
 y1[0]=0.;
 xdintg_(y,&c[0],y1,&temp,&n);
}

/*****************************************************************************

Two-dimensional  back projection  subroutine
using fast algorithm by Nuttal to calculates the index to data
array for each pixel on the ground in i_box,k_box positioned by i,k
k: azimuth ( 2nd order)
i: range   ( ist order)
The equation for index calculation is
     index= c0+c1*k+c2*k^2 + (c3+c4*k+c5*k^2)*i
        =      d0      +      d1*i
     where c(i) (i=0,5) is a set of coefficients for each box and
         each position in the aperture
     Input:
         float *pixel   pointer to first point of the image
                            2-dimensional area
         float *data    pointer to data array
                   Important: data[0]=0.0
                          data[dat_size-1]=0.0
         long  dat_size size of data buffer
```

## Appendix B

```
        float *c      pointer to six coefficients for
                              index computation
        long k_pix_size   size of the patch (pixels) in k axis
        long i_pix_size   size of the patch (pixels) in i axis
        long k_box    patch number in k axis
        long i_box    patch number in i axis
        long i_box_size   number of patches in i axis
    Output:
        index to data array is calculated and pixel is updated

*****************************************************************************/
#define MAX_K_PIX_SIZE  1000
#define MAX_I_PIX_SIZE  1000

void bp(pixel,data,dat_size,c,k_box_base,k_box,i_box,k_pix_size,
    i_pix_size,i_box_size)

float *pixel;    /* array of pixels of the whole 2-dimentional area      */
float *data;     /* data array                                           */
long dat_size;   /* size of data array                                   */
float *c;        /* pointer to six coefficients for index computation    */
long k_box_base; /* 0 for 1st SC, i*NBOX_AZIMUTH/NO_SUPERCARD for ith SC  */
long k_box;      /* patch number in k axis                               */
long i_box;      /* patch number in i axis                               */
long k_pix_size; /* size of the patch (pixels) in k axis                 */
long i_pix_size; /* size of the patch (pixels) in i axis                 */
long i_box_size; /* number of patches in i axis                          */

{
 void xvclip_();
 void vclip_();
 void xvfx4_();
 void fix4_();
 void xvrmp_();
 void vindex_();
 void vramp_();
 void vgathr_();
 void vadd_();
 void vtabi_();

float findex[MAX_I_PIX_SIZE];   /* data index value                     */
long lindex[MAX_I_PIX_SIZE];    /* data index value (round to integer)  */
float pix_temp[MAX_I_PIX_SIZE]; /* temp buffer for back projection      */
float d0[MAX_K_PIX_SIZE];       /* d0= c0+c1*k+c2*k^2                   */
float d1[MAX_K_PIX_SIZE];       /* d1= c3+c4*k+c5*k^2                   */
```

```
 float *pix_index;  /* pointer to current pixel                    */
 long  pix_index_inc; /* each time k is incremented , pixel pointer is jumped */
 long  k;          /* for k and i loops control                   */
 float maxindex;       /* maximum value for data index                */
 float zero=0.0;
 float one=1.0;
 long i_one=1;
 float temp;

 maxindex=(float)(dat_size-1);

 pix_index=pixel+(k_box-k_box_base)*k_pix_size*i_box_size*
       i_pix_size+i_box*i_pix_size;
                    /* index of first point of the patch   */
                    /* with respect to pixel            */
 pix_index_inc=i_pix_size*i_box_size; /* index increment along k axis */

 poly2(d0,k_pix_size,c);     /* generate array of d0                */
 poly2(d1,k_pix_size,c+3);   /* generate array of d1                */

 for(k=0;k<k_pix_size;k++)
  {

/* generate floating point index vector */
  vramp_(&d0[k],&d1[k],findex,&i_one,&i_pix_size);

/* table look up and interpolate       */
/* this function replaces xvclip(),xvfx4(),vgathr()   */
/*
vtabi_(findex,&i_one,&one,&zero,data,pix_temp,&i_one,&dat_size,&i_pix_size);
*/

/* clip index */
  vclip_(findex,&i_one,&zero,&maxindex,findex,&i_one,&i_pix_size);

/* convert to integer index */
  fix4_(findex,&i_one,lindex,&i_one,&i_pix_size);

/* gather data into temporary buffer */
  vgathr_(data,lindex,&i_one,pix_temp,&i_one,&i_pix_size);
/* update pixel array with data from temporary buffer */
  vadd_(pix_index,&i_one,pix_temp,&i_one,pix_index,&i_one,&i_pix_size);

/* pix_index jumps along k axis  */
  pix_index=pix_index+pix_index_inc;
```

```
  } /* k loop */


} /* subroutine */
```

## B-6. Declare All Global Constants (Focus.h)

```
#define NO_SUPERCARD   3          /* number of supercards     */
#define NPOINT_AZIMUTH  600        /* number of bearing lines  */
#define NPOINT_RADIAL   4095       /* number of radial lines   */
#define NPOINT_APER     2304       /* # of positions in aperture */
#define NPOINT_DATA     2048       /* number of original data points */
#define NPOINT_DATA_INTER  NPOINT_DATA*8 /* number of data points after
                                    interpolated */
#define FILTER_LENGTH   255        /* length of FIR filter     */
#define PI 3.141592654
#define RADAR_HEIGHT  60.          /* radar height in feet     */
#define A_OFSET      -15.0         /* offset angle from center line */
#define A_SPAN       54.0          /* spanning angle of the patch */
#define RMINCENTER   550.          /* range from center position to
                          reference point on tha patch */
#define RCENTER_REF   802.0        /* range (ft) of ref. point from center pos. */
#define THETA_CENTER_REF  -15.0  /* angle (deg)of ref. point from center pos. */
#define SAMPLING_RATE 2.0e09       /* sampling frequency of signal  */
#define SAMPLING_PERIOD 5.0e-10    /* ts=1/fs                       */


#define BOX_SIZE_RADIAL    195
#define BOX_SIZE_AZIMUTH   100
#define NBOX_RADIAL    NPOINT_RADIAL/BOX_SIZE_RADIAL
#define NBOX_AZIMUTH  NPOINT_AZIMUTH/BOX_SIZE_AZIMUTH
#define BOX_BASE_START  2
#define NO_KBOX_PROCESS  3
#define IBOX_START     10
#define NO_IBOX_PROCESS  1

#define SECTION_SIZE 4             /* # of sections for one aperture */
#define COEF_SIZE   6              /* # of coeffs for curve fit   */
#define MAXDEG     3               /* maximum degree for poly. fit */
#define NPOINT_BOXSAMPLE_RADIAL 5   /* # of samples in a box in radial
                          direction                 */
#define NPOINT_BOXSAMPLE_AZIMUTH 5  /* # of samples in azimuth direction
                          for every box             */
#define NPOINT_BOXSAMPLE
```

```
NPOINT_BOXSAMPLE_RADIAL*NPOINT_BOXSAMPLE_AZIMUTH
                    /* # of samples in a box for curve fit */
```

## B-7. Declare All Global Variables (Focus.var)

```
struct complex { float r,i;};          /* define a complex type     */

struct cstype {

/* Data Variables */
float pixel[NPOINT_AZIMUTH/NO_SUPERCARD][NPOINT_RADIAL]; /* array of image*/
float data_temp[NPOINT_DATA]; /* temp buffer for data              */
float data[NPOINT_DATA];              /* original data bufer      */
float filter_coef[NPOINT_DATA_INTER];      /* filter coefficients      */
float data_inter[NPOINT_DATA_INTER+136];   /* interpolated data buffer  */
float tstart_err[NPOINT_APER]; /* tstart-tstart_trunc for each position */
struct complex data_inter_fft[NPOINT_DATA_INTER/2]; /* Real->Complex Forward FFT
                                of data_inter          */
struct complex filter_fft[NPOINT_DATA_INTER/2];   /* Real->Complex Forward FFT
                                of filter_coef         */
float nyquist_filter;       /* value of FFT of filter at Nyquist point */
float nyquist_data;         /* value of FFT of data at Nyquist Point   */
long ham_flag;              /* to indicate wheter to use hamming or not */
float ham_coef[NPOINT_APER]; /* hamming weight coefficients           */


/* Geometry Variables */
float a_ofset;            /* offset angle from the center line      */
float a_span;             /* spanning angle of the patch            */
float d_theta;            /* delta angle                      */
float x,theta;            /* coordinate of a pixel                */
float c_theta,s_theta;       /* cosine and sine of theha            */
float d;                  /* distance from radar to center position
                          positive to the right, neg. to the left */
float rcenter,rmincenter,rmaxcenter; /* range from center position      */
float rcenter_ref;          /* range from center pos. to ref point     */
float theta_center_ref;      /* angle of ref. point from center position */
float r_ref;              /* range from any pos. to ref. point       */
float d_rcenter;            /* sampling distance from center position  */
float dr8;                /* (sampling distance)/8                */
float r,rmin,rmax;          /* range from any position              */
float aper_length;          /* length of aperture                   */
float radar_height;         /* height of radar                      */
float radar_height2;        /* square the height of radar           */
```

## Appendix B

```
/* Radar Variables */
float c;                /* speed of wave                    */
float ts;               /* sampling period of signal        */
float fs;               /* sampling frequency of signal     */

long n_azimuth;             /* # of points in azimuth direction      */
long n_radial;              /* # of points in radial direction       */
long n_aper;                /* # of positions in aperture            */
long pix_size;              /* # of points in pixel array            */
long n_data;                /* number of original data points        */
long n_data_inter;          /* number of interpolated data points    */
long n_data_inter_half;     /* 1/2 # of interpolated data points     */
long n_boxsample;           /* # of samples in a box for curve fit   */
long n_coef;                /* # of coefficients curve fit           */
long n_section;             /* # of sections for one aperture        */
long k_pix_size;            /* # of pixels of a box in azimuth direction*/
long i_pix_size;            /* # of pixels of a box in radial dirction */
long k_box_size;            /* # of boxes in azimuth direction       */
long i_box_size;            /* # of boxes in radial direction        */

/* SSL VAriables */
long supercard_id;          /* to identify board number              */
float f_zero;               /* floating point zero                   */
float f_one;                /* floating point 1                      */
long  i_one;                /* integer 1                             */
long stride;                /* stride for supercard ssl              */
long  ierr;                 /* error message from lib call           */
long dummy;                 /* just a dummy argument pass to SC program */
/* working variables */
long n_position;
float e_index,a_index,error,maxerror,minerror;
long ierr_max,jerr_max,ierr_min,jerr_min;
long column_max,row_max,column_min,row_min;
long ia,ib;
long ipixel,jpixel;
float coefc[COEF_SIZE];  /* c coefficients                           */
float
coefa[SECTION_SIZE][NBOX_RADIAL*NBOX_AZIMUTH][COEF_SIZE][MAXDEG+1];/*
a coef. for each c */
long deg[COEF_SIZE];            /* degree for each coefc        */
long mat_c_stride,findex_c_stride,coef_c_stride;
float tstart_diff[NPOINT_APER];   /* time difference between quantized and */
                        /* unquantized values from trigger to  */
                        /* start data acq. at each position    */
```

```
};
```

## B-8. Initialize All Global Variables on Both Processors (varinit.c)

```
/*****************************************************************************/
/* This subroutine performs the following :                                  */
/*      Initialize both host and Supercard Variables                         */
/* Output:                                                                    */
/*      All variables                                                         */
/*****************************************************************************/
#include "focus.h"
#include "focus.var"

extern struct cstype *cs[NO_SUPERCARD];

focus_varinit()
{
 long j;

 for(j=0;j<NO_SUPERCARD;j++)
  {
  cs[j]->n_azimuth=NPOINT_AZIMUTH;     /* # of points in azi. direction */
  cs[j]->n_radial=NPOINT_RADIAL;       /* # of points in rad. direction */
  cs[j]->pix_size=cs[j]->n_azimuth*cs[j]->n_radial/NO_SUPERCARD;
                                /* Pixel array size */
  cs[j]->n_data=NPOINT_DATA;           /* # of orig. data points       */
  cs[j]->n_data_inter=NPOINT_DATA_INTER;/* # of interpolated. data points*/
  cs[j]->n_data_inter_half=NPOINT_DATA_INTER/2;
                                /* 1/2 # of inter. data points   */
  cs[j]->n_aper=NPOINT_APER;           /* # of points in the aperture   */
  cs[j]->n_coef=COEF_SIZE;             /* # of coeffs for curve fit     */
  cs[j]->n_section=SECTION_SIZE;       /* # of sections for one aperture*/
  cs[j]->i_pix_size=BOX_SIZE_RADIAL;   /* # of pixels of a box in rad. dir.*/
  cs[j]->k_pix_size=BOX_SIZE_AZIMUTH;  /* # of pixels of a box in az. dir. */
  cs[j]->i_box_size=NBOX_RADIAL;       /* # of boxes in radial direction  */
  cs[j]->k_box_size=NBOX_AZIMUTH;      /* # of boxes in azimuth direction */


/* Radar Variables   */
  cs[j]->fs=SAMPLING_RATE;             /* Sampling Frequency           */
  cs[j]->ts=SAMPLING_PERIOD;           /* Sampling period              */
  cs[j]->c=3.e8;                       /* Speed of wave                */
```

## Appendix B

```
/* SSL Variables  */
  cs[j]->f_zero=0.;                /* floating point zero          */
  cs[j]->stride=1;                 /* stride for ssl               */
  cs[j]->f_one=1.0;                /* floating point 1             */
  cs[j]->i_one=1;                  /* integer 1                    */
  cs[j]->mat_c_stride=cs[j]->n_boxsample; /* column stride for mat        */
  cs[j]->findex_c_stride=1;        /* column stride for findex     */
  cs[j]->coef_c_stride=1;          /* column stride for coeff      */
  cs[j]->deg[0]=3;                 /* degree for first coefc       */
  cs[j]->deg[1]=3;
  cs[j]->deg[2]=3;
  cs[j]->deg[3]=3;
  cs[j]->deg[4]=2;
  cs[j]->deg[5]=2;

  }
} /* end focus_varinit */
```

## Distribution

Admnstr
Defns Techl Info Ctr
Attn DTIC-DDA (2 copies)
Cameron Sta Bldg 5
Alexandria VA 22304-6145

Defns Advncd Rsrch Proj Agcy
Attn ASTO D Giglio
3701 N Fairfax Dr
Arlington VA 22203-1714

Defns Intllgnc Agcy
Attn CMO-4 G Andreiev
Attn E Thompson
Central MASINT Ofc
Washington DC 20340-5100

Defns Spply Agcy
Attn P Tomlinsen
1110 N Glebe Rd Ste 400
Arlington VA 22201

DoD
Attn J B Dobsa
9800 Savage Rd
FT Meade MD 20755-6000

Director
DoD-BTI
Attn Dr J R Transue
1901 N Beauregard Ste 380
Alexandria VA 22311

Inst for Defns Anlys
Attn B Crane
Attn J Ralston
1801 N Beauregard Stret
Alexandria VA 22311

Dept Army NVESD
Attn AMSEL-RD-NV-R A Tarbell
Attn AMSEL-RD-NV-RPPO K Willey
FT Monmouth NJ 07703

Belvoir RDE Ctr
Dept Army NVESD
Attn SATBE-N-T B Bernard
Attn SATBE-N-T D Franklin
Attn SATBE-N-T T Broach
10101 Gridley Rd Ste 104
FT Belvoir VA 22060-5806

Dept Army NVESD
Attn AMSEL-RD-NV-GSID T Watts
10221 Burbeck Rd Ste 430
FT Belvoir VA 22060-5806

Hdqtrs
Dept of the Army
Army Spc Prog
Attn SARDSO C Keisar
103 Pentagon
Washington DC 20310-0103

ECAC
Attn M Aasen 11TR1/DQT
Attn C Madison
Attn D Quasny
185 Admiral Cochrane Dr
Annapolis MD 21401

Sp & Terrestrial Comm Dirctrt CECOM RDEC
Attn AMSEL-RD-ST-SE-S J Deewall
FT Monmouth NJ 07703-5208

US Army CECOM Intllgnc/Elect Warfare Dirctrt
Attn AMSEL-RD-IEW-SSP B Bennett
Attn AMSEL-RD-IEW-SSP J Cummins
Attn AMSEL-RD-IEW-SSP R Scruppa
Vint Hill Farm Sta
Warrenton VA 22186-5100

## Distribution (cont'd)

Commander
US Army MICOM
Attn AMSMI-RD-WS-UB D Holder
Attn AMSMI-RD-AS-RA C Krolinger
Attn AMSMI-RD-AS-RA J Loomis
Attn AMSMI-RD-AS-RA R R Smith
Redstone Arsenal AL 35898-8000

US Army Rsrch Ofc
Attn J Harvey
PO Box 12211
Research Triangle Park NJ 27709-2211

Nav Air Warfare Ctr
Attn Code 2154 M Frishbee
Attn Code C27731A M Henderson
Attn Code C29504 D Schriener
1 Administrative Circle
China Lake CA 93555-6001

Nav Postgraduate Schl
Attn Code EC/GL G S Gill
Attn ECE Dept Code EC M Morgan
833 Dryer Rd Rm 437
Monterey CA 93942-5121

Nav Rsrch Lab
Attn Code 5348 P Hansens
Attn Code 5340 E L Mokole
Attn Code 5340 M Steiner
4555 Overlook Ave SW
Washington DC 20375-5336

Naval Rsrch Lab
Attn M Sletten
7204 13th Pl
Takoma Park MD 20912

Naval Rsrch Lab
Attn S N Samaddar
Washington DC 22375

Cmdr
Nav Surfc Weapons Ctr
Attn Code F42 S Leong
Dahlgren VA 22448-5000

NRAD Nav Cmand Cntrl & Ocean
Surveillance Ctr
Attn B Dinger
Attn M Pollac
Attn T Tice
San Diego CA 92152

Cmdr Ofcr
NSWC
Attn G Gaunaurd
Attn P Winters
10901 New Hampshire Ave
Silver Spring MD 20904

Cmdr Ofcr
NSWC
Attn F40 J Cavenaugh
Attn F40 D Kirkpatrick
Dahlgrin VA 22448

Ofc of Nav Rsrch
Attn G D Roy
Arlington VA 22217

Air Force Inst of Techlgy
Attn D Dunn
Attn A J Terzuoli
PO Box 3402
Dayton OH 45401-3402

Air Force Inst of Techlgy
Attn LT D J Wolstenholme
Box 43198 (ENG) 2950 P Stret
Wright Patterson AFB OH 45433-7765

Armstrong Lab
Attn P G Petropoulos AL-OES
8103 15th Stret
Brooks AFB TX 78235

CMTCO
Attn M Douglas
Attn C Gardner
Attn MAJ K Reichl
Attn COL S Williams
1030 S Hwy A1A
Patrick AFB OH 32925-3002

## Distribution (cont'd)

Phillips Lab
Attn PL/WSR C Baum
Attn PL/WSR J Berger
Attn PL/WSR S A Blocher
Attn T S Bowen
Attn C J Buchenauer
Attn R K Marek
Attn S C Mason
3550 Aberdeen Ave SE
Albuquerque NM 87109

Rome Lab
Attn R A Shore RL/ERAS
Attn B Tomasic RL/ERA
Hanscom AFB MA 01731-3010

Rome Lab
Attn F Welker
Attn M Wicks
26 Electronic Pky
Griffis AFB NY 13441-4514

Wright Lab
Attn WL/MNMF K Min
Attn A Sullivan
101 W Eglin Blvd
Eglin AFB FL 32542-6810

Wright Rsrch Dev Ctr
Attn WL/AARM R Koesel
Attn WL/AARM K McCoin
Attn WL/AARM D Muki
Attn WL/AARM J Roman
Attn WL/AARM G Urban
2690 C Stret Bldg 22 Ste 1
Wright Patterson AFB OH 45422-7408

Director
Marine Corps Intllgnc Actvty
Attn MCIAD M Howard
2033 Barnet Ave
Quantico VA 22134-5011

Sandia Natl Lab
Attn B Brock
Attn D Cress
Radar Antenna Dev Code 2343
Albuquerque NM 87185-5800

Sandia Natl Lab
Attn Code 2345 R Hurley
PO Box 5800
Albuquerque NM 87185

Sandia Natl Lab
Attn M Buttram
1153 Sandia Natl Lab
Albuquerque NM 87185-5800

Sandia Natl Lab High Power Electromag Dept
Attn G M Loubriel MS 1153
PO Box 5800
Albuquerque NM 87185-1153

NASA-Goddard
Attn R Wallace Code 480
Greenbelt MD 20771

Natl Inst of Stand & Techlgy
Attn J Horst
Bldg 220 Rm B124
Gaithersburg MD 20399

Natl Inst of Stand & Techlgy
Attn M Kanda
Attn 81307 A Ondrejka
325 Broadway
Boulder CO 80303

Boston Univ Dept of ECS Eng
Attn P R Kotiuga
44 Cummington Stret
Boston MA 02215

Catholic Univ Physics Depart
Attn H Uberall
Washington DC 20064

## Distribution (cont'd)

FAMU/FSU Colg of Engrg
Attn F Gross
PO Box 2175
Tallahassee FL 32316-21785

Univ of FL
Elect Commctn Lab
Attn M Barlett
Attn J Bevington
PO Box 140245
Gainesville FL 32614-0245

GA Institute of Tech
Attn GTRI/ESL/CAD H Engler
Atlanta GA 30332

GA Tech
Attn M P Kesler CRB-600
Attn J G Maloney Centenial Rsrch Bldg/STL
Atlanta GA 30332

GA Tech Schl of Mathematics
Attn O P Bruno
Atlanta GA 30332-0160

Howard Univ
Attn P Alakananda
2300 Sixth St Dept EE
Washington DC 20059

MI State Univ Dept of Electrcl Engrg
Attn Kun-Mu Chen
East Lansing MI 48824

Northeastern Univ ECE Dept
Attn S McKnight
Attn D J McLaughlin
Boston MA 02115

OH State Univ
Attn R Moses
Attn J Young
2015 Neil Ave
Columbus OH 43210-1272

OH State Univ OSU/ESL
Attn E K Walton
1320 Kinnear Rd
Columbus OH 43212-1191

Polytech Univ
Attn L Carin
6 Metrotech Center
Brooklyn NY 11201

Polytechnic Univ Electrcl Engrg Dept/WR1
Attn D M Bolle
Attn S U Pillai
6 Metrotech Center
Brooklyn NY 11201

Polytechnic Univ WRI & Dept of Physics
Attn Mok-Ming Leung
6 Metrotech Center
Brooklyn NY 11201

Purdue Univ
Attn M Melloch
1285 Electrical Eng Bldg
West Lafayette IN 47907-1285

TX A&M Univ Dept of Electrcl Engrg
Attn J C Goswami
College Station TX 77843-3128

Univ of AZ Dept of Electrcl & Cmptr Engrg
Attn D G Dudley
Attn S L Dvorak
Tucson AZ 85721

Univ of CA at Santa Barbara Electrcl & Cmptr Engrg Dept
Attn M Rodwell
Santa Barbara CA 93106

Univ of CA Electrcl Engrg Dept
Attn T Itoh
405 Hilgard Ave 56-125B Engr IV Bldg
Los Angeles CA 90024

## Distribution (cont'd)

Univ of FL Elect Commctn Lab
Attn M Bartlett
Attn J Bevington
Attn J Kurtz
PO Box 140245
Gainesville FL 32614-0245

Univ of IL
Attn W C Chew
1406 W Green Stret
Urbana IL 61801

Univ of MD Dept of Electrcl Engrg
Attn E Funk
Attn C H Lee
College Park MD 20742

Univ of MI Ctr for Ultrafast Optic Sci 2200
Attn J Whitaker
Bonisteel Blvd Rm 1006
Ann Arbor MI 48109-2099

Univ of NM CHTM
Attn K Agi
EECE Bldg Rm 125
Albuquerque NM 87131

Univ of PA Electrcl Engrg Depart
Attn R Bose
Attn D Carlson
Attn B D Steinberg
200 S 33rd Stret Rm 202
Moore Philadelphia PA 19104

Univ of TX
Attn W Nunnely
PO Box 19380
Arlington TX 76019

Univ of TX at Austin Dept of Electrcl & Cmptr Engrg
Attn H Ling
Austin TX 78712-1084

Univ of VT Dept of Cmptr Sci & Elec Engrg
Attn K E Oughston
Attn P Smith
Burlington VT 05405-0156

Univ of Wash Dept of Electrcl Engrg FT-10
Attn A Ishimaru
Seattle WA 98195

ANRO Engrg Inc
Attn G F Ross
1800 Second Stret Ste 878
Sarasota FL 34236

AT&T Bell Labs
Attn H Lenzing
40 New Stret
Colts Neck NJ 07722

Barth Elect Inc
Attn J E Barth
1300 Wyoming Stret
Boulder City NV 89005

Battell
Attn G Ruck
505 King Ave
Columbus OH 43201

Battelle
Attn MSIN K2-31 D Sheen
PO Box 999
Richland WA 99352

BDM Fed
Attn G R Salo
1801 Randolph Rd SE
Albuquerque NM 87106

Boeing Defns & Sp Grp
Attn D Iverson
Attn B Merle
PO Box 3999 MS 82-36
Seattle WA 98124-2499

## Distribution (cont'd)

Consulant
Attn V Van Lint
1032 Skylark Dr
La Jolla CA 92037

Dynamics Techlgy Inc
Attn S Borchrdt
1555 Wilson Blvd #320
Arlington VA 22209-2405

E Sys
Attn CBN 177 K Robinett
PO Box 6065
Greenville TX 75403-6056

Electromag Capability Anlys Ctr
Attn J Widner
Attn J Word
185 Admiral Cochrane Dr
Annapolis MD 21401

EMA
Attn R Parella
Attn D Steffen
7655 W Mississippi Ave #300
Lakewood CA 80226-4332

Ensco Inc
Attn J Faist
5400 Port Royal Rd
Springfield VA 22151

Ensco Inc
Attn G Beasley
Attn R Kemerait
445 Pineda Ct
Melbourne FL 32940

ERIM
Attn P Lembo
1101 Wilson Blvd #1100
Arlington VA 22209-2043

ERIM
Attn D Ausherman
Attn D Herrick
Attn D Sheen
Attn J Gorman
Attn R Heimiller
Attn S Degraf
Attn T Lewis
PO Box 134001
Ann Arbor MI 48113-4001

Farr Rsrch
Attn E Farr
614 Paseo Delmar NE
Albuquerque NM 87123

GORCA
Attn K Abend
Attn B Fell
PO Box 2325
Cherry Hill NJ 08034-0181

Group
Attn D Sego
Attn B Merele
Box 3999 MS 82-36
Seattle WA 98124-2499

Grumman Aircraft Sys
Attn S Boles
Attn B29-035 D Cermignian
Bethpage NY 11714-3582

HARC
Attn A Blanchard
Attn J Glazner
Attn B Krenek
Attn B Williams
4800 Research Forrest Dr
The Woodlands TX 77381

## Distribution (cont'd)

HP-Hewlett Packard
Attn K Pastel
29 Burlington Mall Rd
Burlington MA 01803

HP-Hewlett Packard
Attn K McMenamin
3701 Koppers Stret
Baltimore MD 21227

HTB Intrntl
Attn J W Luguire
1840 Wilson Blvd #206
Arlington VA 22201

Hughes Mis Comp
Attn G Cooper
Attn L Frazier
AET Ctr Box 1973
Rancho Cucamonga CA 91729-1973

Hughes Aircraft Co
Attn A Wang 600/CISI
PO Box 3310
Los Angeles CA

IBM Rsrch
Attn A Ruehli
PO Box 218
Yorktown Heights NY 10598

ITT Rsrch Inst
Attn H Riggins
1875 Admiral Cochrane Dr
Annapolis MD 21401

Kaman Sci Corp
Attn J Demarest Dikewood Sect
6400 Uptown Blvd NE Ste 300E
Albuquerque NM 87110

Lincoln Lab
Attn S Ayasli
Attn D Blejer
Attn T Bryant
Attn C Frost
Attn C Lee
Attn T Gross
Attn A K McIntosh
Attn M Mirkin
Attn G Morse
Attn L Novak
244 Wood Stret
Lexington MA 02173

Lockheed Arntcl Sys Comp
Attn H B Lorber
Dept 73-E6 Bldg L-8
86 South Cobb Dr
Marietta GA 30063-0649

Loral Defense Sys
Attn B Woody
PO Box 85
Litchfield Park AZ 85340

Soft Machine Resources Inc
Attn M Rofheart
2140 L Stret NW Ste 1007
Washington DC 20037

Mirage Sys
Attn D Barrick
Attn J Erwin
Attn G Mousally
Attn M Rosowski
Attn P Failer
Attn D Venters
425 Lakeside Dr
Sunnyvale CA 94086

## Distribution (cont'd)

Modern Technologies Corp
Attn B Williams
494 Sycamore Ave
Shrewsbury NJ 07702

Mssn Rsrch Corp
Attn E K English
3975 Research Blvd
Dayton OH 45430

Phys RPI
Attn Xi-Cheng Zhang
Troy NY 12180

Power Spectra Inc
Attn M Gamble
Attn J Grant
Attn J Oicles
919 Hermosa Ct
Sunnyvale CA 94086-4103

Novel Technologies & Concepts
Attn S Davis
PO Box 726
Almeda CA 94501

Sci Applctn Intrntl Corp
Attn L Manning
1710 Goodridge Dr MS 1-5
McLean VA 22102

Schlumberger-Doll Rsrch
Attn R Gaylor
Attn D R Mariani
Old Quarry Rd
Ridgefield CT 06877

Southwest Rsrch Inst
Attn B Duff
Attn K Stevens
6220 Culebra Rd PO Drawer 28510
San Antonio TX 78228-0510

SRI Intrntl
Attn M Baron
Attn D Bender
Attn D Buseck
Attn K A Dreyer
Attn R Vickers
333 Ravenswood Ave
Menlo Park CA 94025-3493

Technisa
Attn P Nelson
1840 Wilson Blvd #206
Arlington VA 22201

Techlgy Svc Corp
Attn R LeFeure
2950 31st Stret
Santa Monica CA 09405-3093

Texas Instr
Attn G Burnham
Attn R Vos
PO Box 405 M/S 3451
Lewisville TX 75067

The Boeing Comp
Attn B H Merle MS-8721
Attn D Iverson
Attn D Sego
PO Box 3999 MS 82-36
Seattle WA 98124-2499

CSPI
Attn J Warther
40 Linnell Circle
Billerica MA 01821

Thermo Trex Corp
Attn C C Phillips
9550 Distribution Blvd
San Diego CA 92121-2306

## Distribution (cont'd)

Toyon
Attn M Vanblaricum
75 Aero Camino Ste A
Goleta CA 93117-3139

Unisys Corp Gov Sys Grp
Attn A Kerdock
Attn K S Lee M/S H16
365 Lakeville Rd
Great Neck NY 11020-1696

United Technologies Norden Sys
Attn M Greenspan
Norden Pl Box 5300
Norwalk CT 06856

Army Rsrch Lab
Attn AMSRL-EP-ED W R Buchwald
Attn M Weiner
FT Monmouth NJ 07703

Army Rsrch Lab
Attn J Arthur
Attn AMSRL-SL-EP J B Henderson
White Sands Missile Range NM 88002-5513

Army Rsrch Lab/EPSD
Attn AMSRL-EP-EC A A Kim
Attn E Lenzing
FT Monmouth NJ 07703

Army Rsrch Lab
Attn AMSRL-EP-EC L Kingsley
Pulse Power Center
FT Monmouth NJ 07703-5601

US Army Rsrch Lab
Attn AMSRL-EP-MC A Paolella
FT Monmouth NJ 07703-5601

US Army Rsrch Lab
Attn AMSRL-OP-SD-TA Mail & Records Mgmt
Attn AMSRL-OP-SD-TL Tech Library (3 copies)
Attn AMSRL-OP-SD-TP Tech Pub
Attn AMSRL-SS J Sattler
Attn AMSRL-SS V DeMonte
Attn AMSRL-SS-FG R Tobin
Attn AMSRL-SS-IB E Adler
Attn AMSRL-SS-S J Miller
Attn AMSRL-SS-SA M Barnes
Attn AMSRL-SS-SA F H Le
Attn AMSRL-SS-SD C N Tran
Attn AMSRL-SS-SG P Alexander
Attn AMSRL-SS-SG A A Bencivenga
Attn AMSRL-SS-SG M Bennett
Attn AMSRL-SS-SG L Happ
Attn AMSRL-SS-SG R Innocenti
Attn AMSRL-SS-SG K Kappra
Attn AMSRL-SS-SG H Khatri
Attn AMSRL-SS-SG L Nguyen (10 copies)
Attn AMSRL-SS-SG M Ressler
Attn AMSRL-SS-SG V Sabio
Attn AMSRL-SS-SG J Sichina (10 copies)
Attn AMSRL-SS-SG T Ton
Attn AMSRL-SS-SJ G H Stolovy
Attn AMSRL-SS-SM R Kapoor
Attn AMSRL-WT-NF L Jasper
Attn AMSRL-WT-NH L Libelo
Attn AMSRL-SS-SG J McCorkle (30 copies)



☆ U.S. GOVERNMENT PRINTING OFFICE: 1994 -- 379-161

POC - Kamie Murray 10106
X 5427

# Quality Assurance and OPSEC Review

This form is an approval record for ARL generated information to be presented or disseminated external to ARL. Note: Submit all manuscripts in electronic format or camera ready copy. See attached instructions. If more space is needed, use reverse of form *(include block numbers)*.

**A. General Information**

1. Present Date (mm/dd/yyyy): 01/29/2010

2. Unclassified Title: ARL-TR-355 Focusing of Dispersive Targets Using Synthetic Aperture Radar

3. Author(s): *(Last, First, MI)*: John McCorkle, Lam Nguyen

4. Office Symbol(s): RDRL-SER-U

5. Telephone Nr(s): (301) 394 - 0847 (LAM NGUYEN)

6. Contractor generated ☒ No ☐ Yes
*If yes, enter Contract No. and ARL COR*

7. Type: ☒ Report ☐ Abstract ☐ Publication ☐ Presentation *(speech, briefing, video clip, poster, etc)* ☐ Book ☐ Book Chapter ☐ Web

8. Key Words: UWB, Synthetic Paerture Radar (SAR), Back Projection

9. Distribution Statement *(required)* is manuscript subject to export control? ☐ No ☐ Yes

Check appropriate letter and number *(see instructions for statement text)*
A ☒ B ☐ C ☐ D ☐ E ☐ F ☐ X ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐

10. Security Classification: UNCLASSIFIE

**B. Reports**

| 11. Series | 12. Type | 13. No. of pages | 14. Project No. | 15. Period Covered | 16. Sponsor |
|---|---|---|---|---|---|
| TR | | | | | |

**C. Publications**

17. Is MS an invited paper? ☒ No ☐ Yes

18. Publication is a refereed journal? ☒ No ☐ Yes

19. Material will be submitted for publication in
Change Distribution to Public Release — Journal
ARL TR — Country

**D. Presentations**

20. Conference Name / Location: ARL TR

21. Sponsor: ARL

22. Conference Date

23. Due Date

24. Conference is ☐ Open to general public ☐ Unclassified / Controlled access ☐ Classified

25. For nonpublic meetings: Will foreign nationals be attending?
☐ No ☐ Yes *(If yes, list countries and identify International Agreement(s))*
☐ Don't know

26. Material will be
☐ Oral presented only ☐ Oral presented and published in
(If published, complete block 18 and 19, Section C).

**E. Authors Statement:** 27. All authors have concurred in the technical content and the sequence of authors. All authors have made a substantial contribution to the manuscript and all authors who have made a substantial contribution are identified in Block 3.

ARL Lead Author or COR — 1/29/2010 Date (mm/dd/yyyy)

**F. Approvals**

28. First line Supervisor of Senior ARL Author or COR.
Name (Last, First, MI) — 01/29/10 Date (mm/dd/yyyy)

29. Reviewer(s) *(Technical / Editorial / NA)*
Name(s) — 01/29/2010 Date (mm/dd/yyyy)

30. Limited distribution information for release to foreign nationals.
Foreign Disclosure — Date (mm/dd/yyyy)

31. Classified Information
Classified by:
Declassified on:
Command Security Manager — Date (mm/dd/yyyy)

**ARL Form 1**
**October 2003**

# OPSEC REVIEW CHECKLIST

**OPSEC POC: Complete and explain any positive responses in block 9.**
**Note: ARL must be the proponent of the proposed information for release.**

1. Does this material contain Sensitive Information? ☐ YES ☒ NO
2. Does this information contain state-of-the-art, breakthrough technology? ☐ YES ☒ NO
3. Does the United States hold a significant lead time in this technology? ☐ YES ☒ NO
4. Does this information reveal aspects of reverse engineering? ☐ YES ☒ NO
5. Does this material reveal any security practices or procedures? ☐ YES ☒ NO
6. Does this information reveal any security practices or procedures? ☐ YES ☒ NO
7. Would release of this information be of economic benefit to a foreign entity, adversary, or allow for the development of countermeasures to the system or technology? ☐ YES ☒ NO
8. Does this material contain:
   a. Any contract proposals, bids, and/or proprietary information? ☐ YES ☒ NO
   b. Any information on inventions/patent application for which patent secrecy orders have been issued? ☐ YES ☒ NO
   c. Any weapon systems/component test results? ☐ YES ☒ NO
   d. Any ARL-originated studies or after action reports containing advice and recommendations? ☐ YES ☒ NO
   e. Weakness and/or vulnerability information? ☐ YES ☒ NO
   f. Any information on countermeasures? ☐ YES ☒ NO
   g. Any fielding/test schedule information? ☐ YES ☒ NO
   h. Any Force Protection, Homeland Defense *(security)* information? ☐ YES ☒ NO
   i. Information on subjects of potential controversy among military services or other federal agencies? ☐ YES ☒ NO
   j. Information on military applications in space, nuclear chemical or biological efforts: high energy laser information; particle beam technology; etc? ☐ YES ☒ NO
   k. Contain information with foreign policy or foreign relations implications? ☐ YES ☒ NO

## OPSEC Approval Statement

I, the undersigned, am aware of the adversary's interest in DOD publications and in the subject matter of this material and that, to the best of my knowledge, the net benefit of this release out weights the potential damage to the essential security of all ARL, AMC, Army, or other DOD programs of which I am aware.

John Costanza

OPSEC Reviewer *(Printed name/signature)* — Date

**9. Space for explanations/continuations/OPSEC review comments**

Since this report was published 16 years ago, the technology described herein has become routine and is no longer critical technology.

## Final Release Clearances

32. Public/Limited release information

a. Material has been reviewed for OPSEC policy.

ARL OPSEC Officer — Date: 25 Feb 10

b. The information contained in this material is ☑ / is not ☐ approved for public release/ has received appropriate tech/editorial review.

for Paul Amirtharaj, Division Chief — Date: 9 Mar 10

c. This information is accepted for public release.

Public Affairs Office — Date