REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-02-

0036

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE

3. REPORT TYPE AND DATES COVERED
01 Jul 98 - 31 Dec 01

4. TITLE AND SUBTITLE
Advanced Mathematics for Missile Seeker Signal Processing

5. FUNDING NUMBERS
F49620-98-C-0034

6. AUTHOR(S)
Dr. Dennis Braunreiter

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Raytheon
Electronic Systems
Missile Systems
P.O. Box 11337
Tucson, AZ 85734-1337

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
801 N. Randolph Street Room 732
Arlington, VA 22203-1977

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-98-C-0034

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF TRANSMITTAL DTIC. THIS TECHNICAL REPORT
HAS BEEN REVIEWED AND IS APPROVED FOR PUBLIC RELEASE
LAW AFR 190-12. DISTRIBUTION IS UNLIMITED.

13. ABSTRACT *(Maximum 200 words)*

This final report summarizes progress toward accomplishment of contractual objectives during the 42-month period covering 01 July 1998 through 31 December 2001. While somewhat arbitrary, it is convenient to break the contract into two periods of performance. We refer to the first period of performance as the Base Program. It covered 33 months from 01 July 1998 to 31 March 2001 and had a funding level of $2,400,000. The Base Program concentrated on radar signal processing and had an objective of using advanced mathematics to improve radar adaptive array processing (AAP). We refer to the second period of performance as the Option Program. It covered 9 months from 01 April 2001 to 31 December 2001 and had a funding level of $640,000. The Option Program concentrated on optical signal processing and had an objective of applying advanced mathematics algorithms developed by various contractors under DARPA DSO ACMP to uncooled JR (UCIR) sensors.

14. SUBJECT TERMS

20020206 094

15. NUMBER OF PAGES
86

16. PRICE CODE

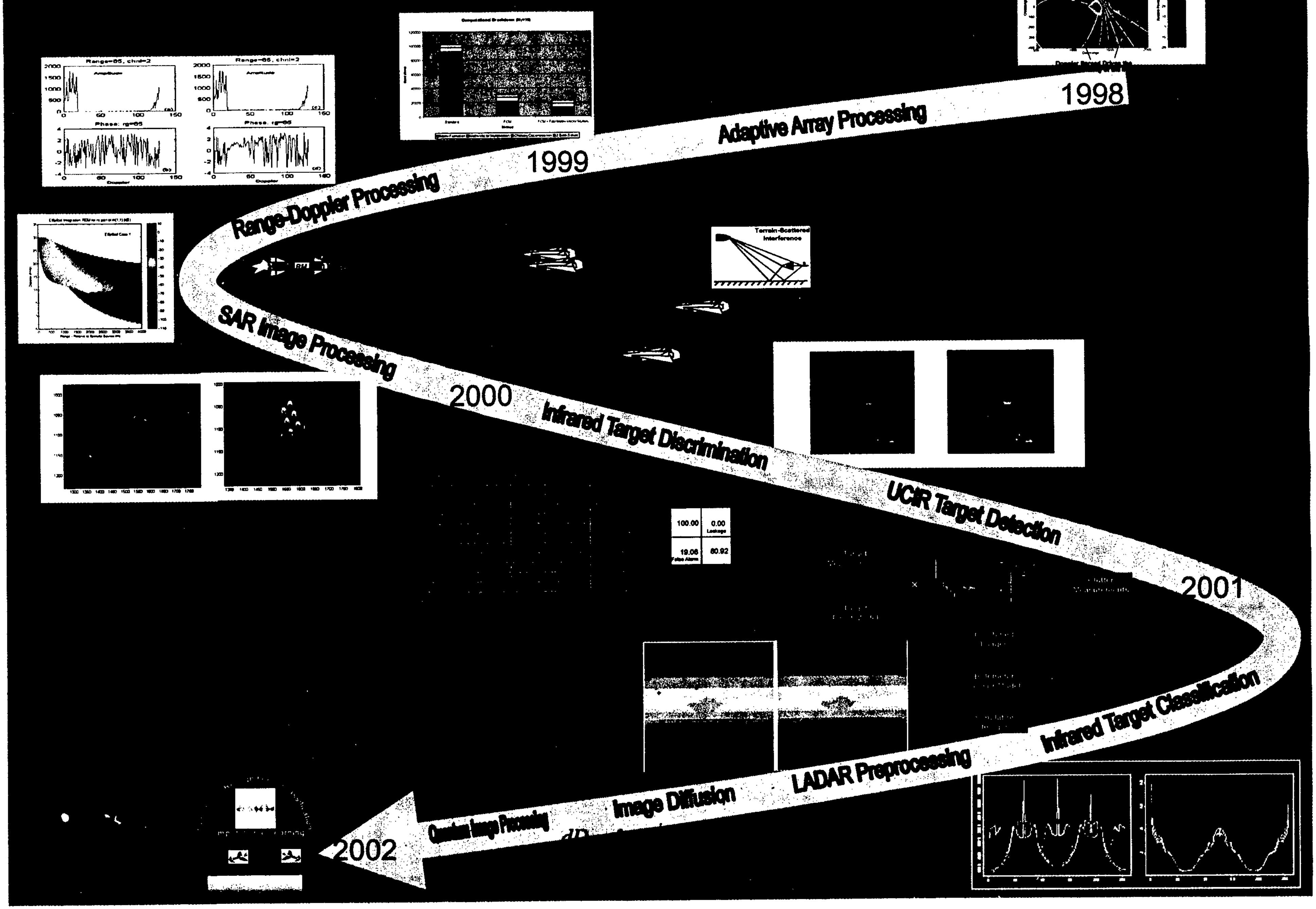
17. SECURITY CLASSIFICATION OF REPORT

18. SECURITY CLASSIFICATION OF THIS PAGE

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT

ADVANCED MATHEMATICS FOR MISSILE SEEKER SIGNAL PROCESSING



1998 Adaptive Array Processing

1999 Range-Doppler Processing

SAR Image Processing

2000 Infrared Target Discrimination

UCIR Target Detection

2001 Infrared Target Classification

LADAR Preprocessing

Image Diffusion

Quantum Image Processing

2002

Raytheon
Raytheon Systems Company

20 December 2001

Progress Report

CLIN No. 0001AA

Final Report for Period of Performance

01 July 1998 - 31 December 2001

Advanced Mathematics for Optimizing Missile Seeker Signal Processing

Program Manager: Dr. Harry A. Schmitt

Principal Investigator: Dr. Harry A. Schmitt

Sponsored By:*
Defense Advanced Projects Agency/DSO
Dr. Douglas Cochran/DARPA DSO
Program Manager: Dr. Arje Nachman/AFOSR
Issued by AFOSR under Contract # F49620-98-C-0034

Prepared By:
Raytheon Systems Company
P.O. Box 11337
Tucson, AZ 85734

Distribution Statement: Approved for public release; distribution is unlimited.

TABLE OF CONTENTS

Index of Figures

Index of Tables

Advanced Mathematics for Optimizing Missile Seeker Signal Processing
CLIN No. 0001AA: Final Report for F49620-98-C-0034

0.0 Program Introduction

This final report summarizes progress toward accomplishment of contractual objectives during the 42-month period covering 01 July 1998 through 31 December 2001. While somewhat arbitrary, it is convenient to break the contract into two periods of performance. We refer to the first period of performance as the Base Program. It covered 33 months from 01 July 1998 to 31 March 2001 and had a funding level of $2,400,000. The Base Program concentrated on radar signal processing and had an objective of using advanced mathematics to improve radar adaptive array processing (AAP). We refer to the second period of performance as the Option Program. It covered 9 months from 01 April 2001 to 31 December 2001 and had a funding level of $640,000. The Option Program concentrated on optical signal processing and had an objective of applying advanced mathematics algorithms developed by various contractors under DARPA DSO ACMP to uncooled IR (UCIR) sensors.

We believe that overall this has been a highly successful program. We have developed and/or evaluated a number of algorithms for both radar and infrared signal processing. These include:

- Base Program
  - Multiresolution Space Time Adaptive Processing
  - Fast Covariance Matrix Formation
  - Fast Matrix-Vector Multiplication
  - Range-Doppler Phase Detection
  - Duke University Channel Equalization
- Option Program
  - Local Singular Value Decomposition
  - Anisotropic Image Diffusion
  - Best Basis Template Matching
  - Multiresolution Anisotropic Image Diffusion
  - Duke University improvements to the Local Singular Value Decomposition

In addition, we have had two significant program transitions and have a very high probability of achieving a third.

- Fast Channel Equalization algorithm transitioned to AMRAAM Phase 3
- Fast Range-Doppler Map generation algorithm transitioned to Raytheon's Air-to-Air 6 Degree of Freedom simulation
- Probable transition of the Local Singular Value Decomposition algorithm to the NetFires program

We believe that we have managed the program well. While our staffing levels have fluctuated throughout the program, we have managed to remain on schedule and within budget. Moreover, we have been able to maintain a core technical team, which has been quite challenging given the staffing shortages Raytheon has experience over the last few years. We have managed to get and keep subcontracts in place with a number of universities. We have supported Drs. Healy and Cochran by responding to all technical and programmatic requests in a timely manner.

1.0 RF Adaptive Array Processing

1.1 Introduction and Background

Received signals may contain components due to target, receiver noise, and unwanted interference (clutter, jamming, or both). Nonadaptive processing is the typical mode for processing these signals; however, when a threat is encountered that requires interference nulling or spectrum estimation, adaptive processing will be required. The adaptive processing mode is assumed to be ABF to cancel interference when the interference is separable from the target in range or Doppler, and SSE when the signal and the target are inseparable in range or Doppler. Figure 1a illustrates a tactical scenario requiring AAP, while Figure 1b illustrates the antenna beam patterns for ABF and SSE.

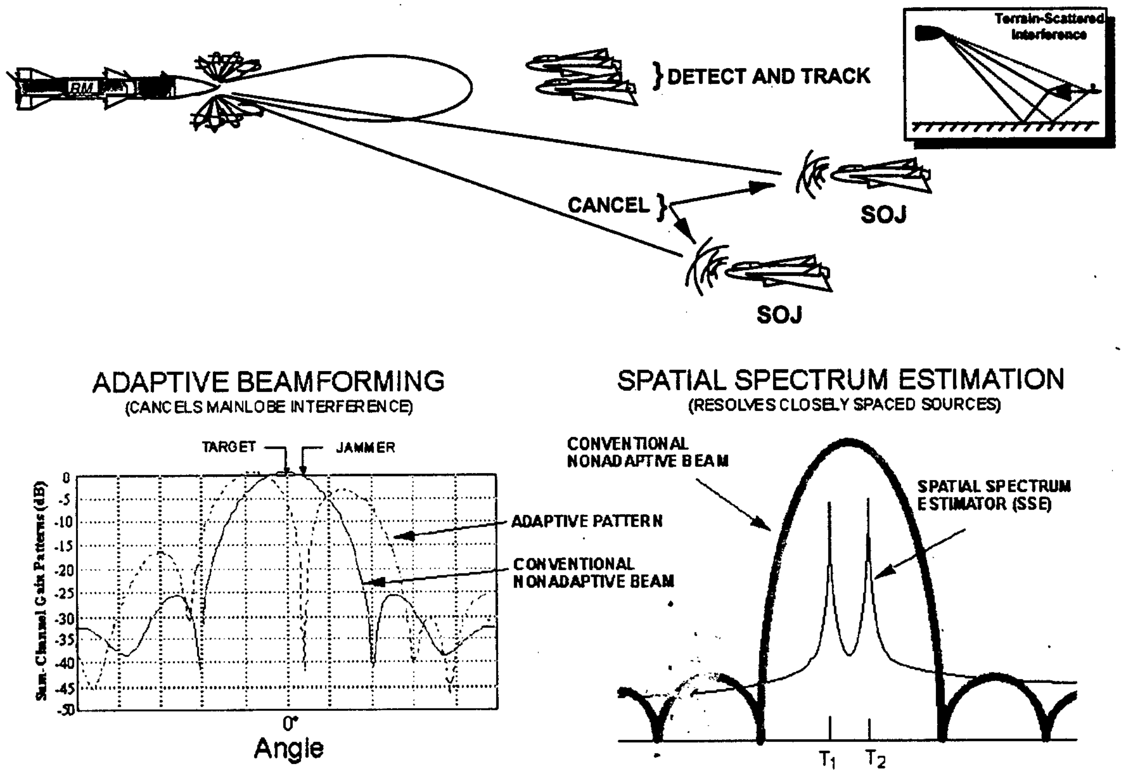
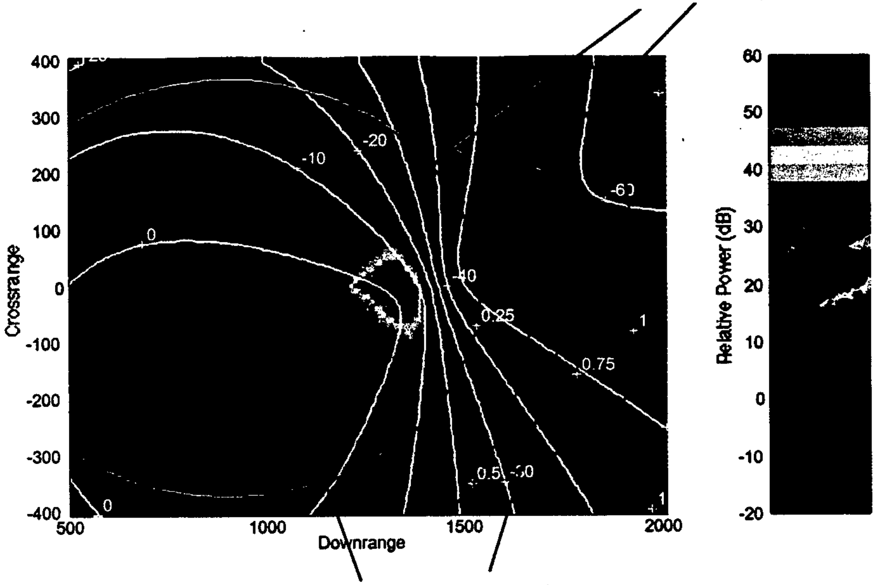


Figure 1: (a) Tactical AAP Scenario; (b) Adaptive Processing - ABF and SSE

Broadly speaking, the Terrain Scattered Interference (TSI) environments important to the radar processing problem are narrowband clutter and a wideband, jamming environment. While the narrowband clutter is a significant problem in target detection, the jamming environment is what drives the high throughput requirement. We will use a community accepted MIT Lincoln Labs Airborne Seeker Testbed (ASTB) Model to model the TSI. This model has been developed from flight test data taken from an airborne platform employing jamming techniques. In Figure 2, we show a typical TSI Clutter Isorange/IsoDoppler Map, which reveals strong non-stationary components. It is this highly non-stationary environment that necessitates either high dimensional Coherent Processing Interval (CPI) rate processing or fast update, lower dimension Pulse Repetition Interval (PRI) rate processing.



Non-Adaptive Sum Channel Power, typical TBJ scenario

Delay Spread

Doppler Spread Drives the Non-stationarity of the Interference

Figure 2: TSI Non-Stationary Interference

Space Time Adaptive Array Processing (STAP) is a digital processing technique that attempts to exploit the information present in the available degrees of freedom (DOF). The usual challenge for adaptive array processing in air-to-air missile radar seekers is the detection and radar-parameter estimation of a nonradiating, low Signal-to-Noise Ratio (SNR) target return (reflection of seeker illumination waveform) in the presence of much higher Jammer-to-Noise Ratio (JNR), wide-band-noise interference. The most demanding Electronic Countermeasure (ECM) scenario is associated with low altitude targets and on-board or off-board ECM that produce, deliberately or inadvertently, terrain-scattered interference (TSI). The ability to suppress this interference to sufficiently low levels for achieving the required Signal-to-Interference Noise Ratio (SINR), is embodied in the estimated spatio-temporal covariance matrix.

Adaptive processing attempts to remove the interference to extract obscured targets. STAP works by estimating the covariance matrices from sample data that does not contain the target. As shown in Figure 3, a set of weight vectors, w_n, is found from the sample covariance matrix, Λ, in addition to a set of steering vectors, s_n. Specifically, we need to solve the matrix equation $w_n = \Lambda^{-1}s_n$ and it is this numerically intensive process that drives the STAP throughput requirements. In practice, STAP calculations are often performed in the data (voltage) domain.

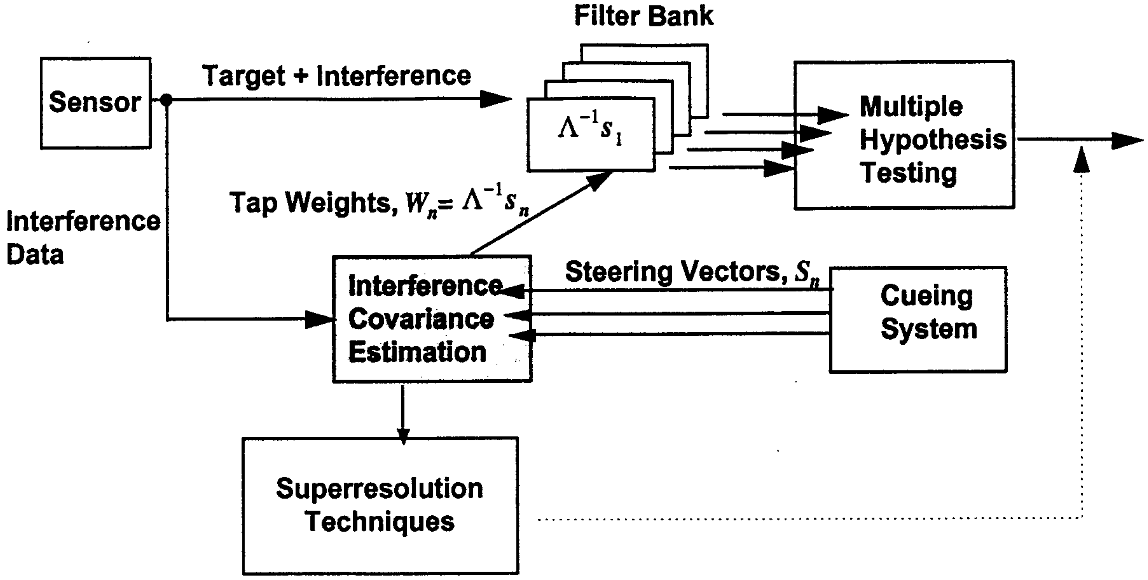


Figure 3: Adaptive Processing Schematic

Figure 4 shows the overall functional architecture considered for this research study. This adaptive processing architecture forms the basis for throughput estimation, as well as identifying those areas that may benefit the most by new processing techniques. From this figure, we can identify the critical adaptive processing functions: (i) channel equalization; (ii) covariance matrix and data matrix operations; and (iii) adaptive array processing, such as ABF and SSE.

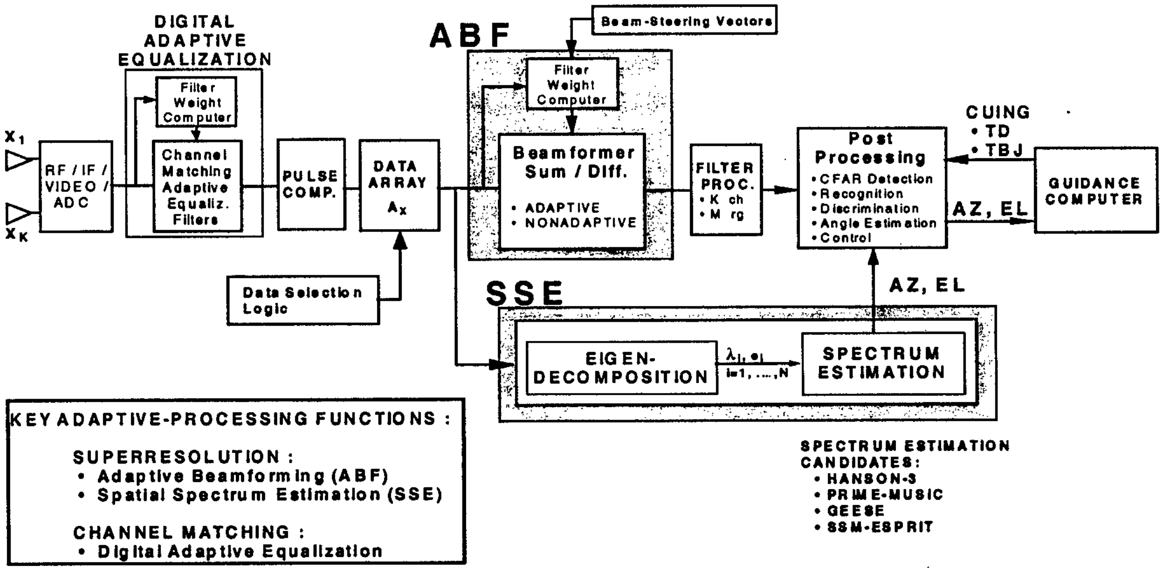


Figure 4: Adaptive Processing Architecture

1.2 Fast Channel Equalization Algorithm

Future Air-to-Air and Surface-to-Air missiles are expected to require RF Adaptive Array Processing (AAP) for Adaptive Beam Forming (interference nulling) or Spatial

Spectrum Estimation (superresolution) to counter advanced Electronic Counter Measures. Critical to the performance of these adaptive processing algorithms is interchannel equalization, since adaptive processing null depth is limited by the equalization filter performance. In addition, as RF bandwidth increases to accommodate improved range resolution, the number of filter taps required to achieve a given level of equalization performance grows.

In this section, we describe a new channel equalization algorithm, the Fast Channel Equalization (FCE) algorithm, for interchannel equalization and RF AAP that is significantly faster than conventional methods. As an example, FCE is five times faster for a sixteen-tap FIR filter. For future threats, which may require up to 128 tap filters, FCE is more than ten times faster. FCE was developed by Professor Gregory Beylkin of the University of Colorado and was funded under the current contract. Several other development programs are considering the FCM algorithm for RF AAP.

1.2.1 Algorithm Description

As we mentioned earlier, the problem that we need to solve is $\Lambda w_n = X^H s_n$. Here $\Lambda \equiv X^H X$ where X is the data matrix and the superscript H indicates the complex conjugate transpose. This problem is conveniently partitioned into four pieces: (i) formation of Λ; (ii) matrix-vector multiplication $X^H s_n$; (iii) Cholesky decomposition; and (iv) two back solves. For missile systems, the general dimensions of the problem are that Λ is about *16x16* and X is about *80x16*. The relatively small matrix sizes involved mean that asymptotic approaches are unlikely to yield significant computational savings. In fact, for a sixteen-tap FIR filter, the formation of the covariance matrix is by far the dominant factor, followed by the matrix-vector multiplication.

The Fast Covariance Matrix (FCM) algorithm for matrix formation and multiplication is based on the identification of a Hankel symmetry present in the data matrix, X, that is used to construct the covariance matrix. While the Hankel symmetry is evident by looking at the form of the data matrix illustrated in Figure 5, calculating the displacement rank of the covariance matrix originally identified it. Most previous approaches had looked for symmetries present in the covariance matrix. While the covariance matrix is approximately Toeplitz, the symmetry is not exact. A forcing the Toeplitz symmetry on the covariance matrix by zero-padding the data matrix, for example, leads to an unacceptable ~3 dB loss in performance.

$$A=\begin{bmatrix} y_1 & y_2 & \cdot\;\cdot\;\cdot & y_{N_T-1} & y_{N_T} \\ y_2 & y_3 & \cdot\;\cdot\;\cdot & y_{N_T} & y_{N_T+1} \\ y_3 & y_4 & \cdot\;\cdot\;\cdot & y_{N_T+1} & y_{N_T+2} \\ \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot \\ y_{M_1-2} & y_{M_1-1} & \cdot\;\cdot\;\cdot & y_{M_1+N_T-4} & y_{M_1+N_T-3} \\ y_{M_1-1} & y_{M_1} & \cdot\;\cdot\;\cdot & y_{M_1+N_T-3} & y_{M_1+N_T-2} \\ y_{M_1} & y_{M_1+1} & \cdot\;\cdot\;\cdot & y_{M_1+N_T-2} & y_{M_1+N_T-1} \end{bmatrix} \Lambda \propto \begin{bmatrix} \sum_{i=1}^{M_1} y_i^* y_i & \sum_{i=1}^{M_1} y_i^* y_{i+1} & \cdot\;\cdot\;\cdot & \sum_{i=1}^{M_1} y_i^* y_{i+N_T-2} & \sum_{i=1}^{M_1} y_i^* y_{i+N_T-1} \\ \sum_{i=1}^{M_1} y_{i+1}^* y_i & \sum_{i=1}^{M_1} y_{i+1}^* y_{i+1} & \cdot\;\cdot\;\cdot & \sum_{i=1}^{M_1} y_{i+1}^* y_{i+N_T-2} & \sum_{i=1}^{M_1} y_{i+1}^* y_{i+N_T-1} \\ \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot \\ \sum_{i=1}^{M_1} y_{i+N_T-2}^* y_i & \sum_{i=1}^{M_1} y_{i+N_T-2}^* y_{i+1} & \cdot\;\cdot\;\cdot & \sum_{i=1}^{M_1} y_{i+N_T-2}^* y_{i+N_T-2} & \sum_{i=1}^{M_1} y_{i+N_T-2}^* y_{i+N_T-1} \\ \sum_{i=1}^{M_1} y_{i+N_T-1}^* y_i & \sum_{i=1}^{M_1} y_{i+N_T-1}^* y_{i+1} & \cdot\;\cdot\;\cdot & \sum_{i=1}^{M_1} y_{i+N_T-1}^* y_{i+N_T-2} & \sum_{i=1}^{M_1} y_{i+N_T-1}^* y_{i+N_T-1} \end{bmatrix}$$

Figure 5: Structure of Data and Covariance Matrices

Based on the form of the covariance matrix, it is clear that one needs to form only the first row in its entirety. The fact that the covariance matrix is normal means that only the upper triangular portion needs to be calculated. The Hankel symmetry allows most of $\Lambda_{n+1,\,p+1}$ to be calculated from the $\Lambda_{n,\,p}$. These properties can be summarized as:

$$\Lambda_{np} = \sum_{m=1}^{M} y_{n+m-1}^{*} y_{p+m-1} \;;\quad n = 1,\cdots,N \;;\quad p = n,\cdots,N \;;$$

$$\Lambda_{pn} = \Lambda_{np}^{*} \quad n = 1,\cdots,N-1 \;;\quad p = n+1,\cdots,N \,.$$

$$\Lambda_{n+1\,p+1} = \Lambda_{np} - y_n^{*} y_p + y_{n+M}^{*} y_{p+M} \;;\quad n = 1,\cdots,N-1 \;;\quad p = n,\cdots,N-1$$

The matrix-vector multiplication $X^H s_n$ can be carried out more efficiently by manipulating the standard fast algorithm for multiplying a square Toeplitz matrix by a vector. We reffer to this algorithm as the Fast Matrix-Vector Multiplication (FMVM) algorithm. It is clear that the product of a Toeplitz matrix and a vector is trivially related to the product of a Hankel matrix and a vector:

$$Mu = Nv; \quad N = Ms; \quad v = s^{-1}u; \qquad s = \begin{bmatrix} 0 & 0 & . & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ . & . & . & . & . \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

1.2.2 Timing Results

Because of the relative simplicity of the FCE algorithm, it is possible to directly calculate the number of floating point operations required and compare this to the standard approach. As shown in Figure 6, the computational saving are impressive, especially for the small size of the matrices involved. The largest computational component, the covariance matrix formation, has been reduced by a factor of eight, while the overall computational load has been reduced by a factor of five.

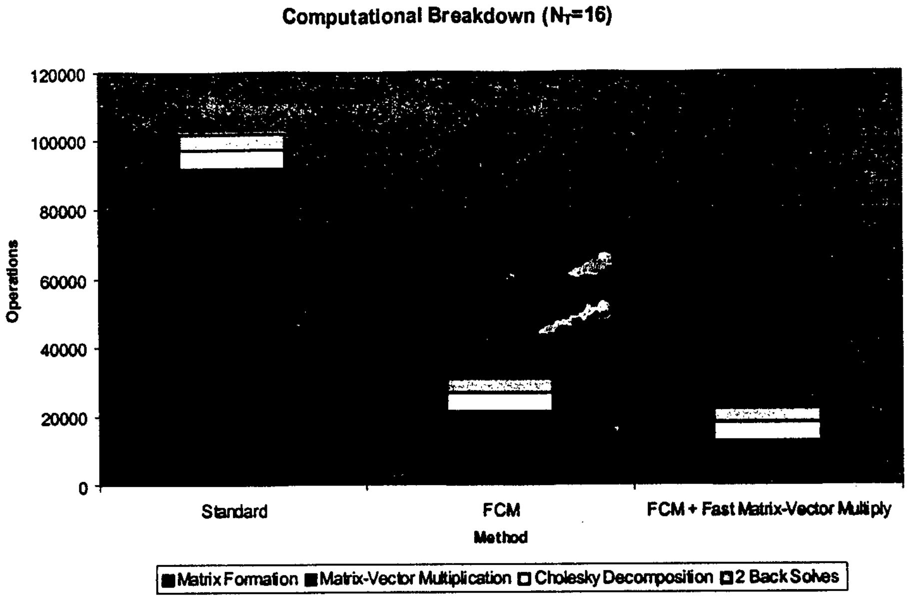


Figure 6: Computational Breakdown for 16 Tap FIR Filter

Future RF missiles are expected to require up to 128 tap FIR filters. The computational gains for the FCE are even more impressive here; these are shown in Figure 7.

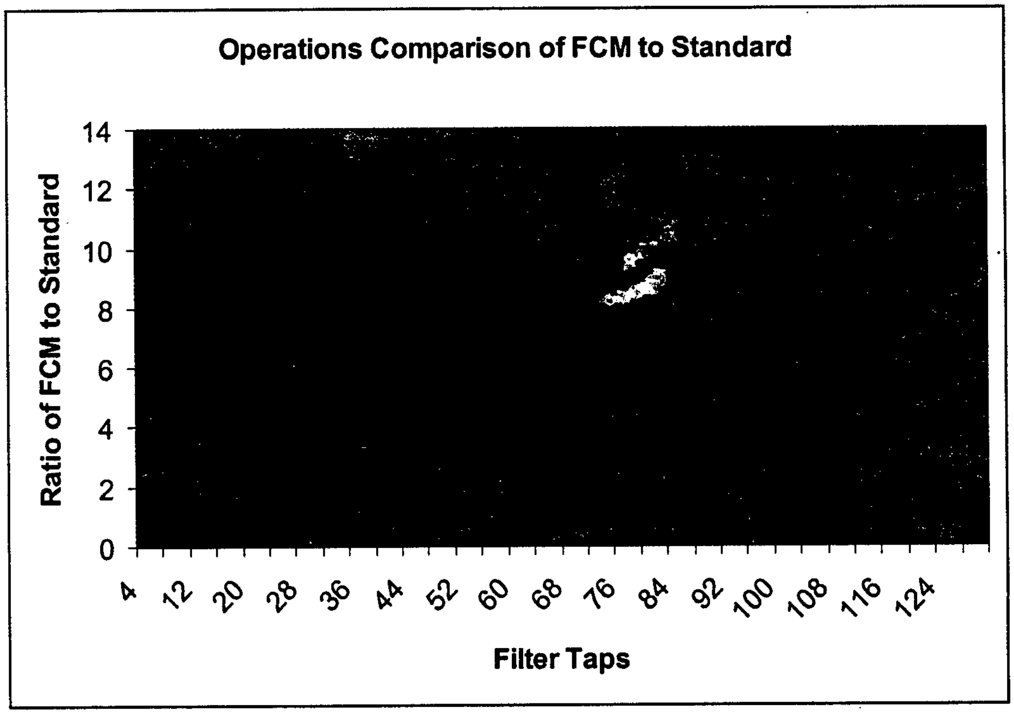


Figure 7: Processing Gains for Future RF Systems

1.2.3 Duke University Channel Equalization (DUCE) Algorithm

This section describes a channel equalization algorithm developed by Professor Xiaobai Sun from Duke University. While the work was not funded under the current contract, it is directly related to our program and was motivated by results we presented at ACMP PI meetings. We start again with the data and covariance matrices:

$$\mathbf{X}=\begin{pmatrix} X_1 & X_2 & X_3 & \cdots & X_{N-2} & X_{N-1} & X_N \\ X_2 & X_3 & X_4 & \cdots & X_{N-1} & X_N & X_{N+1} \\ X_3 & X_4 & X_5 & \cdots & X_N & X_{N+1} & X_{N+2} \\ \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots & \vdots \\ X_{M-2} & X_{M-1} & X_M & \cdots & X_{M+N-5} & X_{M+N-4} & X_{M+N-3} \\ X_{M-1} & X_M & X_{M+1} & \cdots & X_{M+N-4} & X_{M+N-3} & X_{M+N-2} \\ X_M & X_{M+1} & X_{M+2} & \cdots & X_{M+N-3} & X_{M+N-2} & X_{M+N-1} \end{pmatrix}; \quad \Lambda = \mathbf{X}^H\mathbf{X}$$

We note that each member of the set $\{X_N\, X_{N+1} \cdots X_{M-1}\, X_M\}$ contributes to every element of the covariance matrix. The remaining elements are the sequences $X^{head} = \{X_1\, X_2 \cdots X_{N-2}\, X_{N-1}\}$ and $X^{tail} = \{X_{M+1}\, X_{M+2} \cdots X_{M+N-2}\, X_{M+N-1}\}$. The DUCE algorithm forms the Cholesky factorization of the covariance matrix $\Lambda = \mathbf{L}\mathbf{L}^H$. $\mathbf{L}$ is a lower triangular matrix and requires as input only the first column of the covariance matrix and X^{head}, X^{tail}. The desired N by N lower triangular matrix $\mathbf{L}$ is first set to all zeros and then the first column is defined in the usual way:

$$L_{11} = \sqrt{A_{11}} \qquad\qquad L_{n1} = \frac{A_{n1}}{L_{11}}\,;\ n = 2,3,\ldots,N\,.$$

In addition, we define

$$\mathbf{F}=\begin{pmatrix} F_{11} & 0 & 0 & 0 \\ F_{21} & F_{22} & F_{23} & F_{24} \\ F_{31} & F_{32} & F_{33} & F_{34} \\ F_{41} & F_{42} & F_{43} & F_{44} \\ \vdots & \vdots & \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,2} & F_{N-1\,3} & F_{N-1\,4} \\ F_{N\,1} & F_{N\,2} & F_{N\,3} & F_{N\,4} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & X_1 & L_{21} & X_{M+1} \\ L_{31} & X_2 & L_{31} & X_{M+2} \\ L_{41} & X_3 & L_{41} & X_{M+3} \\ \vdots & \vdots & \vdots & \vdots \\ L_{N-1\,1} & X_{N-2} & L_{N-1\,1} & X_{M+N-2} \\ L_{N1} & X_{N-1} & L_{N1} & X_{M+N-1} \end{pmatrix}.$$

Note that the first column is initially the same as the first column of $\mathbf{L}$. The third column of $\mathbf{F}$ is also initially the same as the first column of $\mathbf{L}$, except that the first element of the latter is 0. The 2nd through N^{th} elements of the second and fourth columns are respectively identical to the sequences X^{head} and X^{tail}.

The algorithm to evaluate $\mathbf{L}$ that will be described below involves looping $N-1$ times for $k=2,\ldots,N$. Each such loop results in the evaluation of the elements in a new column of $\mathbf{L}$. After having completed loop $k-1$, $\mathbf{L}$ and $\mathbf{F}$ have the form

$$\mathbf{L}=\begin{pmatrix} L_{11} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ L_{21} & L_{22} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ L_{31} & L_{32} & L_{33} & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots \\ L_{k-2\,1} & L_{k-2\,2} & L_{k-2\,3} & \cdots & L_{k-2\,k-2} & 0 & 0 & \cdots & 0 & 0 \\ L_{k-1\,1} & L_{k-1\,2} & L_{k-1\,3} & \cdots & L_{k-1\,k-2} & L_{k-1\,k-1} & 0 & \cdots & 0 & 0 \\ L_{k\,1} & L_{k\,2} & L_{k\,3} & \cdots & L_{k\,k-2} & L_{k\,k-1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots \\ L_{N-1\,1} & L_{N-1\,2} & L_{N-1\,3} & \cdots & L_{N-1\,k-2} & L_{N-1\,k-1} & 0 & \cdots & 0 & 0 \\ L_{N\,1} & L_{N\,2} & L_{N\,3} & \cdots & L_{N\,k-2} & L_{N\,k-1} & 0 & \cdots & 0 & 0 \end{pmatrix}; \quad \mathbf{F}=\begin{pmatrix} F_{11} & 0 & 0 & 0 \\ F_{21} & 0 & 0 & 0 \\ F_{31} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ F_{k-2\,1} & 0 & 0 & 0 \\ F_{k-1\,1} & 0 & 0 & 0 \\ F_{k\,1} & F_{k\,2} & F_{k\,3} & F_{k\,4} \\ F_{k+1\,1} & F_{k+1\,2} & F_{k+1\,3} & F_{k+1\,4} \\ \vdots & \vdots & \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,2} & F_{N-1\,3} & F_{N-1\,4} \\ F_{N\,1} & F_{N\,2} & F_{N\,3} & F_{N\,4} \end{pmatrix}$$

The first $k-2$ elements of the first column are equal to the first $k-2$ diagonal elements of $\mathbf{L}$ and the remaining elements of that column are equal to the non-zero elements of column $k-1$ of $\mathbf{L}$. The non-zero elements in the last three columns of $\mathbf{F}$ are other combinations of the elements of the original $\mathbf{F}$. At the start of loop k each of the last $N-k+1$ elements of $\mathbf{F}$ is replaced by the numerical value in the element above it (a downward shift of elements) and the other elements of $\mathbf{F}$ remain the same.

The next step in loop k is to define parameters

$$\rho_1=\sqrt{|F_{k\,1}|^2+|F_{k\,2}|^2}; \quad \rho_2=\sqrt{|F_{k\,3}|^2+|F_{k\,4}|^2}$$
$$\beta_1=\frac{F_{k\,1}}{\rho_1}; \quad \gamma_1=\frac{F_{k\,2}}{\rho_1}; \quad \beta_2=\frac{F_{k\,3}}{\rho_2}; \quad \gamma_2=\frac{F_{k\,4}}{\rho_2}.$$

The matrices formed from these parameters are unitary and represent complex-two dimensional rotations.

$$\mathbf{U}=\begin{pmatrix} \beta_1^* & -\gamma_1 \\ \gamma_1^* & \beta_1 \end{pmatrix}; \quad \mathbf{V}=\begin{pmatrix} \beta_2^* & -\gamma_2 \\ \gamma_2^* & \beta_2 \end{pmatrix}; \quad \left(\mathbf{U}^H=\mathbf{U}^{-1};\mathbf{V}^H=\mathbf{V}^{-1}\right)$$

The last $N-k+1$ elements of the first two columns of $\mathbf{F}$ are now transformed with $\mathbf{U}$ and the last $N-k+1$ elements of the last two columns of $\mathbf{F}$ with $\mathbf{V}$ as follows:

$$\begin{pmatrix} F_{k\,1} & F_{k\,2} \\ F_{k+1\,1} & F_{k+1\,2} \\ F_{k+2\,1} & F_{k+2\,2} \\ \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,2} \\ F_{N\,1} & F_{N\,2} \end{pmatrix} \leftarrow \begin{pmatrix} F_{k\,1} & F_{k\,2} \\ F_{k+1\,1} & F_{k+1\,2} \\ F_{k+2\,1} & F_{k+2\,2} \\ \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,2} \\ F_{N\,1} & F_{N\,2} \end{pmatrix} \begin{pmatrix} \beta_1^* & -\gamma_1 \\ \gamma_1^* & \beta_1 \end{pmatrix},$$

$$\begin{pmatrix} F_{k\,3} & F_{k\,3} \\ F_{k+1\,3} & F_{k+1\,4} \\ F_{k+2\,3} & F_{k+2\,4} \\ \vdots & \vdots \\ F_{N-1\,3} & F_{N-1\,4} \\ F_{N\,3} & F_{N\,4} \end{pmatrix} \leftarrow \begin{pmatrix} F_{k\,3} & F_{k\,3} \\ F_{k+1\,3} & F_{k+1\,4} \\ F_{k+2\,3} & F_{k+2\,4} \\ \vdots & \vdots \\ F_{N-1\,3} & F_{N-1\,4} \\ F_{N\,3} & F_{N\,4} \end{pmatrix} \begin{pmatrix} \beta_2^* & -\gamma_2 \\ \gamma_2^* & \beta_2 \end{pmatrix}.$$

It follows that under this transformation $F_{k\,1} \leftarrow \rho_1$; $F_{k\,2} \leftarrow 0$; $F_{k\,3} \leftarrow \rho_2$; $F_{k\,4} \leftarrow 0$, and, at this point in loop k, $\mathbf{F}$ has the form

$$\mathbf{F} = \begin{pmatrix} F_{11} & 0 & 0 & 0 \\ F_{21} & 0 & 0 & 0 \\ F_{31} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ F_{k-2\,1} & 0 & 0 & 0 \\ F_{k-1\,1} & 0 & 0 & 0 \\ F_{k\,1} & 0 & F_{k\,3} & 0 \\ F_{k+1\,1} & F_{k+1\,2} & F_{k+1\,3} & F_{k+1\,4} \\ \vdots & \vdots & \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,2} & F_{N-1\,3} & F_{N-1\,4} \\ F_{N\,1} & F_{N\,2} & F_{N\,3} & F_{N\,4} \end{pmatrix}.$$

Next in loop k the following parameters are calculated

$$\rho_3 = \sqrt{|\rho_1|^2 - |\rho_2|^2}; \quad \beta_3 = \frac{\rho_1}{\rho_3}; \quad \gamma_3 = \frac{\rho_2}{\rho_3},$$

The matrix formed from these parameters is a hyperbolic rotation with the property:

$$\mathbf{W} = \begin{pmatrix} \beta_3^* & -\gamma_3 \\ -\gamma_3^* & \beta_3 \end{pmatrix}; \quad \mathbf{W}^{-1} = \begin{pmatrix} \beta_3 & \gamma_3 \\ \gamma_3^* & \beta_3^* \end{pmatrix},$$

The last $N-k+1$ elements of the first and third columns of $\mathbf{F}$ are now transformed with $\mathbf{W}$ as follows:

$$\begin{pmatrix} F_{k\,1} & F_{k\,3} \\ F_{k+1\,1} & F_{k+1\,3} \\ F_{k+2\,1} & F_{k+2\,3} \\ \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,3} \\ F_{N\,1} & F_{N\,3} \end{pmatrix} \leftarrow \begin{pmatrix} F_{k\,1} & F_{k\,3} \\ F_{k+1\,1} & F_{k+1\,3} \\ F_{k+2\,1} & F_{k+2\,3} \\ \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,3} \\ F_{N\,1} & F_{N\,3} \end{pmatrix} \begin{pmatrix} \beta_3^* & -\gamma_3 \\ -\gamma_3^* & \beta_3 \end{pmatrix}.$$

It follows that under this transformation $F_{k\,1} \leftarrow \rho_3;\quad F_{k\,3} \leftarrow 0$. The form that $\mathbf{F}$ has at the end of loop k is

$$\mathbf{F} = \begin{pmatrix} F_{11} & 0 & 0 & 0 \\ F_{21} & 0 & 0 & 0 \\ F_{31} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ F_{k-1\,1} & 0 & 0 & 0 \\ F_{k\,1} & 0 & 0 & 0 \\ F_{k+1\,1} & F_{k+1\,2} & F_{k+1\,3} & F_{k+1\,4} \\ F_{k+2\,1} & F_{k+2\,2} & F_{k+2\,3} & F_{k+2\,4} \\ \vdots & \vdots & \vdots & \vdots \\ F_{N-1\,1} & F_{N-1\,2} & F_{N-1\,3} & F_{N-1\,4} \\ F_{N\,1} & F_{N\,2} & F_{N\,3} & F_{N\,4} \end{pmatrix}.$$

The first $k-1$ elements of the first column are equal to the first $k-1$ diagonal elements of $\mathbf{L}$ and the remaining elements of that column are equal to the non-zero elements of column k of $\mathbf{L}$. At the end of the loop k, $\mathbf{L}$ has the form:

$$\mathbf{L} = \begin{pmatrix} L_{11} & 0 & 0 & \cdots & 0 & 0 & 0 \cdots 0\ 0 \\ L_{21} & L_{22} & 0 & \cdots & 0 & 0 & 0 \cdots 0\ 0 \\ L_{31} & L_{32} & L_{33} & \cdots & 0 & 0 & 0 \cdots 0\ 0 \\ \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots & \vdots\ \vdots\vdots\vdots\ \vdots\ \vdots \\ L_{k-2\,1} & L_{k-2\,2} & L_{k-2\,3} & \cdots & 0 & 0 & 0 \cdots 0\ 0 \\ L_{k-1\,1} & L_{k-1\,2} & L_{k-1\,3} & \cdots & L_{k-1\,k-1} & 0 & 0 \cdots 0\ 0 \\ L_{k\,1} & L_{k\,2} & L_{k\,3} & \cdots & L_{k\,k-1} & L_{k\,k} & 0 \cdots 0\ 0 \\ \vdots & \vdots & \vdots & \vdots\vdots\vdots & \vdots & \vdots & \vdots\ \vdots\vdots\vdots\ \vdots\ \vdots \\ L_{N-1\,1} & L_{N-1\,2} & L_{N-1\,3} & \cdots & L_{N-1\,k-1} & L_{N-1\,k} & 0 \cdots 0\ 0 \\ L_{N\,1} & L_{N\,2} & L_{N\,3} & \cdots & L_{N\,k-1} & L_{N\,k} & 0 \cdots 0\ 0 \end{pmatrix}.$$

We have evaluated the performance of the DUCE algorithm on the sixteen-tap FIR filter problem. Timing studies indicate that the DUCE algorithm is approximately twice as fast as the FCM algorithm for the same system. Because the covariance matrix is never calculated directly, we expect that this algorithm will perform even better on embedded code, where creating and populating a matrix can be a significant percentage of the overall time.

1.2.4 Extension of FCM to Space Time Adaptive Processing (STAP)

There is a straightforward extension of the covariance matrix formation of the FCM algorithm to STAP. In this case the data matrix possess a block Hankel structure and matrices replace the scalar matrix elements of the covariance matrix. This is illustrated in Figure 8. Here N_T corresponds to the number of time taps and $M = 5N_T$. We note that the computation reduction for covariance matrix formation is of the order of N_T. This is the same as that achieved by enforcing a block Toeplitz structure on the covariance matrix but without the ~3 dB performance loss.

$$A = \begin{bmatrix} \vec{y}_1 & \vec{y}_2 & \cdots & \vec{y}_{N_T-1} & \vec{y}_{N_T} \\ \vec{y}_2 & \vec{y}_3 & \cdots & \vec{y}_{N_T} & \vec{y}_{N_T+1} \\ \vec{y}_3 & \vec{y}_4 & \cdots & \vec{y}_{N_T+1} & \vec{y}_{N_T+2} \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \vec{y}_{M_1-2} & \vec{y}_{M_1-1} & \cdots & \vec{y}_{M_1+N_T-4} & \vec{y}_{M_1+N_T-3} \\ \vec{y}_{M_1-1} & \vec{y}_{M_1} & \cdots & \vec{y}_{M_1+N_T-3} & \vec{y}_{M_1+N_T-2} \\ \vec{y}_{M_1} & \vec{y}_{M_1+1} & \cdots & \vec{y}_{M_1+N_T-2} & \vec{y}_{M_1+N_T-1} \end{bmatrix}$$

$$\Lambda \propto \begin{bmatrix} \sum_{i=1}^{M_1} \vec{y}_i^H \vec{y}_i & \sum_{i=1}^{M_1} \vec{y}_i^H \vec{y}_{i+1} & \cdots & \sum_{i=1}^{M_1} \vec{y}_i^H \vec{y}_{i+N_T-2} & \sum_{i=1}^{M_1} \vec{y}_i^H \vec{y}_{i+N_T-1} \\ \sum_{i=1}^{M_1} \vec{y}_{i+1}^H \vec{y}_i & \sum_{i=1}^{M_1} \vec{y}_{i+1}^H \vec{y}_{i+1} & \cdots & \sum_{i=1}^{M_1} \vec{y}_{i+1}^H \vec{y}_{i+N_T-2} & \sum_{i=1}^{M_1} \vec{y}_{i+1}^H \vec{y}_{i+N_T-1} \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \sum_{i=1}^{M_1} \vec{y}_{i+N_T-2}^H \vec{y}_i & \sum_{i=1}^{M_1} \vec{y}_{i+N_T-2}^H y_{i+1} & \cdots & \sum_{i=1}^{M_1} \vec{y}_{i+N_T-2}^H \vec{y}_{i+N_T-2} & \sum_{i=1}^{M_1} \vec{y}_{i+N_T-2}^H \vec{y}_{i+N_T-1} \\ \sum_{i=1}^{M_1} \vec{y}_{i+N_T-1}^H \vec{y}_i & \sum_{i=1}^{M_1} \vec{y}_{i+N_T-1}^H \vec{y}_{i+1} & \cdots & \sum_{i=1}^{M_1} \vec{y}_{i+N_T-1}^H \vec{y}_{i+N_T-2} & \sum_{i=1}^{M_1} \vec{y}_{i+N_T-1}^H \vec{y}_{i+N_T-1} \end{bmatrix}$$

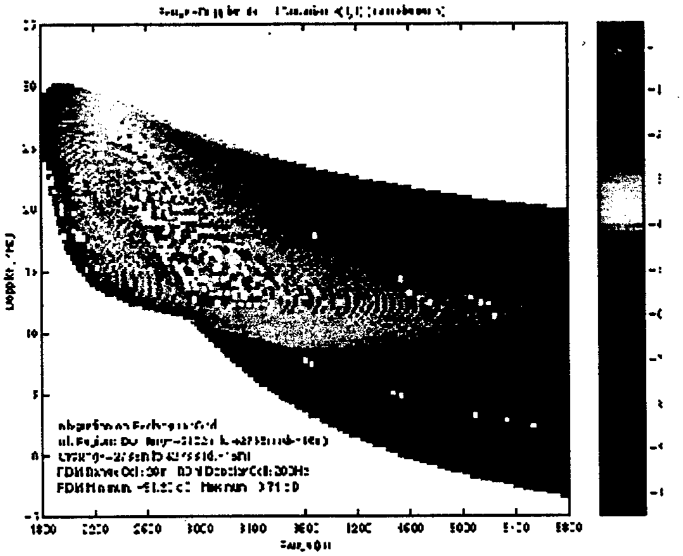
Figure 8: Structure of Data and Covariance Matrices

1.3 Efficient Algorithm for Range-Doppler Map Generation

We have developed a highly efficient algorithm to generate range-Doppler maps. These maps are a critical component of Raytheon's high fidelity simulations and 6 Degree-of-Freedom (6 DOF) models. The new algorithm is 10x faster than the current method and produces superior results – fewer artifacts due to quantization, better localization capability, *etc.* While the original motivation for developing this algorithm was to study the effect of correlated random scattering on adaptive processing performance, there is also significant program interest in this new algorithm. This program interest is driven by the government's request to evaluate several new flight trajectories that require the generation of significantly larger range-Doppler maps. The current algorithm is far too slow to be implemented in the 6-DOF. Current timing estimates verify that the elliptical approach provides a ten-fold reduction in computational run times.

To illustrate the higher quality results of the elliptical integration approach, we generate a Range-Doppler map with the rectangular integration simulation. The purpose of this coarse grid run is to illustrate the "holes" in the range-Doppler map that result from the rectangular integration method. These holes are due to the fact that uniform sampling in down range and cross range results in a non-uniform sampling in the bounce-range plane. The plot of where the rectangular grid evaluation points are mapped into the

Range-Doppler plane illustrates the non-uniform sampling. The consequence of this effect is that finer sampling is required when using the rectangular integration method. Finer sampling, however, requires more simulation run time, and possibly multiple runs due to array size limitations. This is the reason that the baseline fine grid run with the rectangular integration method used to validate the elliptical integration method required eight runs to cover the same ground area as a single elliptical integration run.



1.

Figure 9: Range-Doppler Map Generated with Rectangular Grid

Completing the comparison, we display (in Figure 10) a range-Doppler map generated with the elliptical grid approach. Not only does it remove the shortcomings of the rectangular grid approach, the results achieved are obtained with only 10% of the computational effort.

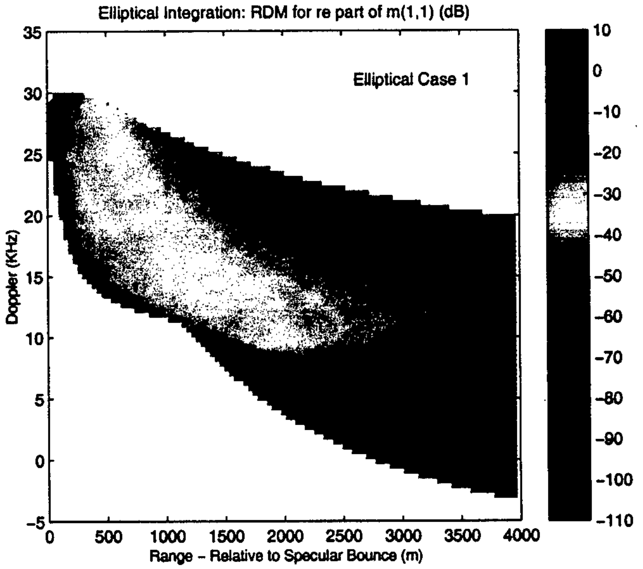


Figure 10: Range-Doppler Map Generated with an Elliptical Grid

The grid shapes for the rectangular and elliptical integration regions are illustrated in Figures 11(a) and 11(b), respectively. The evaluation points for the ground maps are at the intersections of the dotted lines. For the rectangular integration method, the evaluation points are at the intersections of constant downrange and crossrange lines. For the elliptical integration method, the evaluation points are at equal parametric-angle intervals on constant bounce range ellipses.

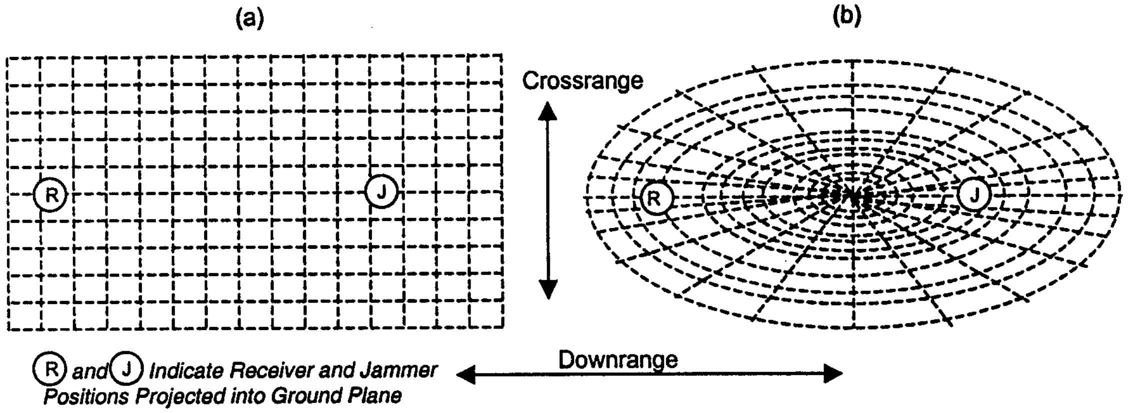


Figure 11: Grid Shapes for Rectangular & Elliptical Integration

1.3.1 Elliptical Grid Integration

The rationale behind using the new elliptical integration approach for computing the range-Doppler maps is multi-faceted and pragmatic: namely, it produces maps that are smoother and more accurate than the previously used rectangular integration approach—and it does this about 10 times more quickly.

Assuming a flat earth scenario, it is readily shown that radar signal paths corresponding to a fixed distance single-bounce path (jammer-ground-receiver) trace out ellipses on the ground. If the ground bounce point is coincident with the specular point, then the total bounce path is minimal and the ellipse is degenerate, *i.e.*, a single point. Single bounce paths of greater length trace out progressively larger ellipses on the ground. These ellipses are not concentric and they have varying degrees of eccentricity. But, given a known set of coordinates for the jammer and receiver, and a specified excess bounce path distance (amount above specular distance), the equations for these ground ellipses are all computable in closed form. In other words, the isorange contours on the earth's surface representing constant bounce range are analytically known, relatively simple functions (ellipses).

The contours corresponding to lines of constant Doppler shift ("isodops"), however, are not represented as simply as the isorange ellipses. The equations which they satisfy are of the form:

$$\text{Doppler Shift} = \quad (\mathbf{v}_R \cdot \mathbf{r}_R)\,/\,|\,\mathbf{r}_R \cdot \mathbf{r}_R\,| + (\mathbf{v}_J \cdot \mathbf{r}_J)\,/\,|\,\mathbf{r}_J \cdot \mathbf{r}_J\,|$$

where $\mathbf{v}_R$ and $\mathbf{v}_J$ are, respectively, the velocity vectors of the receiver and jammer, and $\mathbf{r}_R$ and $\mathbf{r}_J$ are, respectively, the location vectors of the receiver and jammer relative to a point (x,y) on the ground. Expanding this equation (using much algebra) to remove the square roots of the right-hand side denominators, one finally obtains an eighth-degree polynomial relating "x" and "y". The locus of (x,y) points for a given value of Doppler Shift is an "isodop" curve, *i.e.*, a line of constant Doppler. Since the relationship between x and y is, in general, an 8th degree polynomial, there are no closed form solutions for x in terms of y, nor y in terms of x (unlike the situation arising in the case of the expressions for isorange contours).

1.3.2 Approximations

In setting up the bins for the range-Doppler maps, bin boundaries for range and Doppler are defined to lie along contours of constant range and Doppler. Each bin is bounded by an upper and lower isorange contour and by upper and lower isodops. The isorange contours are handled exactly (to machine precision) while the isodops are, of necessity, handled approximately. Regarding the isodops, the following steps and assumptions are followed.

First, the basic region of integration on the ground is the "pseudo-triangular" region determined by a single point on the perimeter of one ellipse, connected to the two endpoints of an arc of a second ellipse (see Figure 12). The ellipses are isorange contours. As drawn, the integration region is convex; this is because the single point is located on the inner ellipse. If another example were drawn, one with the single point located on the outer ellipse and connected to two points on the inner ellipse, then the resulting integration region would be concave (with the concavity along the arc length of the inner ellipse). The two ellipses are isorange contours and the two straight lines are a way of breaking up the large isorange region into smaller pieces.

Second, the Doppler shift is computed at each of the three vertices of the pseudo-triangular integration region, using the basic geometric and kinematic information describing the scenario.

Third, to vastly simplify the placement of Doppler shift into bins, one introduces a planar approximation to the Doppler shift over the pseudo-triangular integration region. Using known Doppler shifts computed at the three vertices, coefficients e, f, and g in the equation below are determined:

$$\text{Doppler Shift}(x, y) = e \cdot x + f \cdot y + g$$

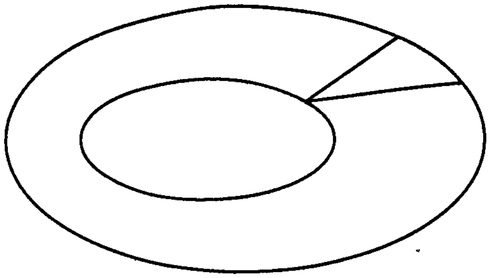


Figure 12: Basic integration region, a "pseudo-triangular" pie slice

Due to this planar approximation of the Doppler velocity, the isodop curves are simply equally spaced parallel lines (see Figure 13). The range-Doppler bin is the area bounded by the two parallel lines (isodops) and the two lines connecting the vertex on the inner ellipse to the two vertices on the outer ellipse. In Figure 13, region B denotes the range-Doppler bin.

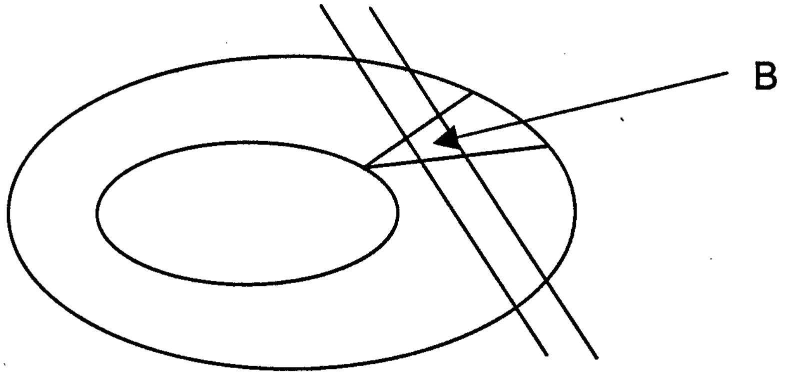


Figure 13: An individual range-Doppler bin

One final step brings the integration region B into an orientation where the double integration is performed more readily. The region is rotated so that the isodop lines are vertically oriented, with velocity-increasing from left to right. The boundaries of region B then consist of, in general, straight lines and elliptical arcs.

Now we address how the integrands are approximated in this numerical approach. All integrands are assumed to take the form of a second degree polynomial in x and y, namely:

$$F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f$$

Functions of this form, when integrated over 2-D regions with boundaries comprised of lines or elliptical arcs, can be integrated in closed form. Equations 2.261 and 2.262-1 through 2.262-3 of Gradshteyn & Ryzhik (1994, *Tables of Integrals, Series and Products*, Academic Press) are used for the integrations that arise when elliptical arc boundaries occur. Those equations are used, respectively, to integrate the following expressions in closed form:

$$\int dx/R^{1/} \qquad \int R^{1/2}dx \qquad \int xR^{1/2}dx \qquad \int x^2R^{1/2}dx,$$

where $R = a + bx + cx^2$.

Looking at the left part of Figure 14 (region ABCD), the six small circles represent the six locations where the integrand function F(x, y) is evaluated along the two elliptical ring arcs. There are three points on each elliptical arc. Knowing the values of x and y at each of the 6 points, plus the values of F(x, y) at those points, a linear system is set up that may be solved for the fitting coefficients: a, b, c, d, e, and f.

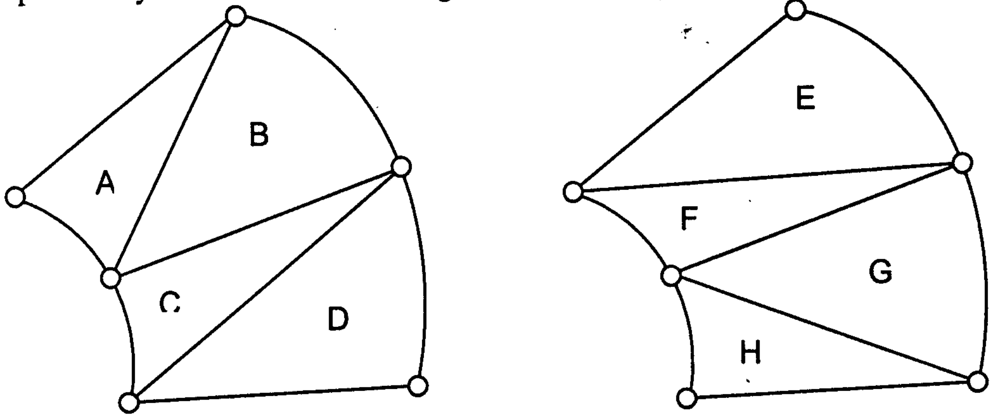


Figure 14: Integrand fitting technique and "opposite parity" accuracy test.

To preserve numerical accuracy, the actual linear system of 6 equations and 6 unknowns is a modified version that has offsets and scale factors applied to the variables x and y. Doing this greatly lowers the condition number of the matrix equation, and hence reduces errors.

The coefficients a, b, c, d, e, and f are then used to approximate the value of the integrand F(x, y) over the entire region covered by subregions A, B, C, and D of Figure 14. (Combined region ABCD is the approximation domain of the six fitting points.) Values of the Doppler velocity at the three vertices of each of the subregions A, B, C, and D are then used in a separate approximation step to "dice up" the individual regions A, B, C, and D into many smaller pieces, delineated by equispaced isodop lines. It is the integrals over these smaller regions that correspond to and contribute to the ultimate result and purpose of the computation—summations of power terms into bins of the range-Doppler map.

As the two parts of Figure 14 show, geometrically identical regions ABCD and EFGH may be subdivided in two different ways. Ideally, regions A and B should contribute to the range-Doppler map in the same way that regions E and F do. (The same statement also follows for regions C and D and regions G and H.) The reality, however, is that slightly different answers are obtained. This is because the isodop lines in the overlap region of A and E, for example, would be different, depending on whether the vertices of region A or the vertices of region E are used to generate the planar approximation to the Doppler velocity in their common region. The amount of disagreement between the range-Doppler maps produced by using the two different "parities" of regional boundaries is an indicator of how well the spacings between elliptical rings and the density of fitting points along the rings represent variations in the Doppler velocity. Closer ring spacings and more points per elliptical ring will naturally tend to yield more accurate planar approximations to the Doppler velocity.

Point selection and fitting procedures for the special case of the central ellipse region are different from the method portrayed in Figure 14. The special case situation is portrayed in Figure 15. The integrand function F(x, y) is fitted over a region consisting of two neighboring pie slices, regions A and B. The six fitting points are: (1) the specular point near the center of the ellipse, (2, 3, and 4) the points along the elliptical arc, and (5 and 6) the points midway between the specular point and the elliptical arc endpoints.

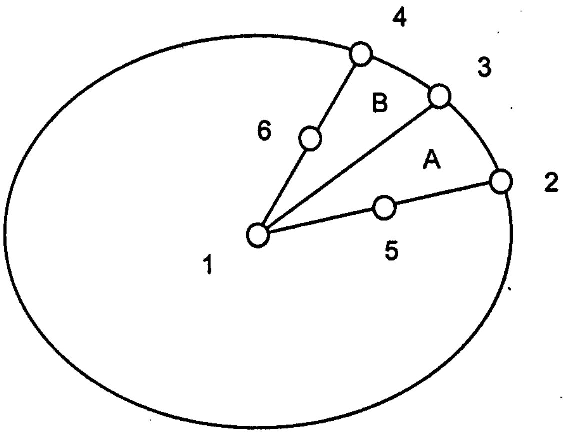


Figure 15: Central ellipse and specular point fitting scenario

Points along the elliptical rings representing constant range may be selected in one of two ways: the "parametric" angle method or the "physical" angle method. The parametric method will always be used for production runs of the software, since it produces more accurate results than the physical method. In the parametric method, the (x, y) points are determined by the equations: $x - x_c = a \cos t$, $y = b \sin t$, where a and b are the semi-major and semi-minor axes of the ellipse, x_c is the centroid location, and t runs over a set of equally spaced parametric angles ranging from 0 to 360 degrees. In the physical method, the (x, y) points are determined by the intersection points of the ellipse with a set of equi-angular rays spreading outward from the ellipse center. In both point selection methods, an even number of points must be selected for distribution around the ellipse perimeters.

There is an exception to the above-described integrand-fitting approach that uses 6 sampling points. If the second-degree polynomial in x and y varies too wildly or rapidly over the fitting region, then unphysical results may sometimes occur. For example, when fitting the diagonal elements of the covariance matrix (which must, theoretically, always have non-negative values), it was found that equation occasionally yielded solutions with negative values of F(x, y) within the fitting region, even though F(x, y) was positive at all six fitting points. To avoid this scenario, an arbitrary test was designed to check for this possibility, followed by imposition of an alternative method for approximating F(x, y), one that guaranteed physically sensible results.

The check for potential troubles was based on finding the critical point of F(x, y). If this critical point was located too close to the triangular region's centroid point, then the second degree polynomial approximation was dropped in favor of using a simple planar fit to F(x, y) at the region's 3 vertices. "Too close" was defined to be within a distance equal to the longest side of the triangle. The critical point of the function F(x, y) is the location of the solution of the pair of simultaneous equations:

$$\partial F/\partial x = 0 \qquad \text{and} \qquad \partial F/\partial y = 0$$

1.3.3 Checks

Given that a wide range of binning options is available via input quantity specifications (e.g., the number of elliptical rings, ring spacing, and number of points per ring), certain accuracy destroying configurations had to be avoided. For instance, if a line drawn from a point on an inner ellipse to a point on an outer ellipse intersects the inner ellipse, certain bookkeeping tasks regarding the region of integration are performed incorrectly. To avoid this possibility, a quick check is performed (before any integrations are started) to confirm that all points on the elliptical rings satisfy this "non-self-intersection" constraint.

1.3.4 Tests

Numerous diagnostic tests of the elliptical integration routine were performed. Cases with independently known, exact answer compared against the output generated by the elliptical integration routine. For example, numerical integrals corresponding to the areas of ellipses were checked; additional integrals corresponding to the centroids and moments of inertia of ellipses were also successfully checked.

1.4 Range-Doppler Phase Detection Algorithm

In conventional monopulse RF seekers, only the amplitudes of the complex-valued range/Doppler-filter outputs are used for target detection. The random-like phases of the range/Doppler-filter output values are seldom considered helpful for target detection. We show that we can achieve robust detection in the phase-domain even when a low-Doppler target has been totally obscured in the amplitude-domain by mainlobe clutter. We have developed several methods to extract phase-information signals for target detection using correlation and spectrum-analysis techniques. Moreover, the phase signals used for target detection can be used for target-direction estimation. Our methods have been tested using by high-fidelity simulation tools and real monostatic clutter data.

The particular problem that we address was the detection of slowly moving or stationary targets in monostatic clutter. Some examples of these types of targets are surface vehicles, launchers and loitering UAVs. The root of the difficulty lies in the fact that conventional radar processing loses the ability to use the target's Doppler to discriminate it from the clutter. Indeed, the target need not even be nearly stationary for this to be a problem – even a rapidly moving target can exhibit low Doppler relative to mainlobe clutter if its velocity vector is nearly perpendicular to the velocity vector of the observation platform.

A representative case where the target signal is totally embedded in strong monostatic clutter was simulated. Nominal values of range-gate resolution and PRF were taken as 40 meters and 33 KHz, respectively. Within each *PRI*, there are 200 contiguous range samples with a range-sample interval of 20 meters, which represents one half of the range-filter resolution. Normal level electronic noise is also included. A point target is located 1.700 Km from the missile with a Doppler frequency of 2.38K Hz and a signal-to-clutter-plus-noise-ratio (SCNR) of -10 dB. Since the interval between Doppler filters is 258 Hz, the major energy of this point target is centered at the range-Doppler cell (85, 9).

1.4.1 Detecting Targets in range-Doppler Phase-Domain

By plotting a cut at range bin 85, we note that the phase curve always has a π-jump at the location of the peak amplitude (d=9). The phase function of a point target in the Doppler-dimension contains strong low frequency components if we perform spectrum analysis of this function; on the other hand, the phase function of a monostatic clutter return generally contains high frequency components. Figure 16 shows the amplitude and phase of a clutter return cut at r=85 in the Doppler-dimension for the cases of monostatic clutter only and target buried in monostatic clutter. The presence of a target is clearly visible by comparing the phase plots.

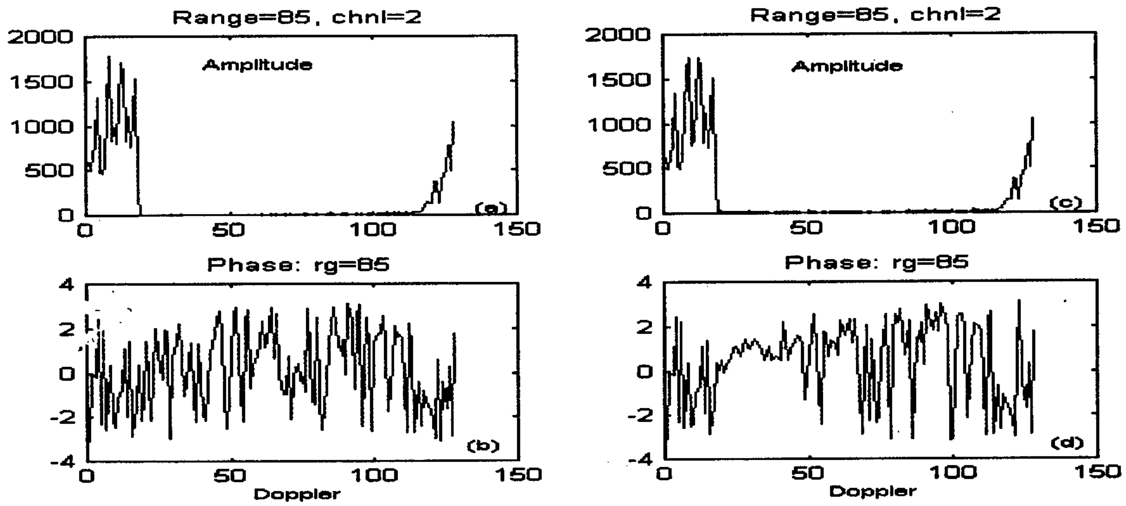


Figure 16: Amplitude and Phase of Clutter Only and Clutter with Target

The algorithm for detecting targets in phase-domain is as follows. The low frequency components are extracted by PSD measurements. Clutter returns and receiver noise are further suppressed by cross-correlation analysis between antenna channels. Two methods are now described for target detection.

Method 1 for Target Detection

For further suppressing clutter and receiver noise, we first cross-correlate the corresponding 1-D slices along the Doppler direction between two space-channels at the same range location:

$$T(r,j)=\sum_{d=0}^{N-1}T_1(r,d+j)T_2(r,d)$$

This operation will suppress clutter and receiver noise since the target signals in different channels have high correlation while the decorrelated clutter and the uncorrelated receiver noise in different channels have much lower correlation. The PSD of the correlated function is then measured. The first several low frequency components are averaged, which further suppresses clutter and receiver noise.

Method 2 for Target Detection

An alternative approach is to calculate the PSDs for different space channels in the phase-domain and then the PSD functions are cross-correlated. A second PSD measurement is taken on the obtained cross-correlation function. The difference between the two methods is that, in one, the original phase functions are used for cross-correlation, while in other, the PSDs of the original phase functions are used for cross-correlation. We have found that the power of the target remains high from low PSD frequency to high PSD frequency, while the power of the clutter and receiver noise does not stretch to higher PSD frequency. Therefore, we can further suppress the clutter and receiver noise by averaging the power along the PSD frequency.

We have also done some preliminary investigation with targets with multiple scatterers and multiple point targets. In this simulation, we used an extended target with two scatterers separated 10 meters along the range direction. The target intensity was centered at r=85 and r=86 with Doppler d=9. RF signals containing multiple point targets separated at nearby range-gates have also been simulated. Based on the previous simulation using a single point target located at r=85, we added another point target located at r=76. Both methods detected the targets for both of these scenarios.

For target detection in the amplitude-domain, the optimal case is that the target Doppler frequency is centered at one of the Doppler filters. Otherwise, the target energy will be picked up by several nearby filters causing "leakage". However, this is not true for target detection in the phase-domain. In general, the optimal Doppler frequency for target detection in the phase-domain is the frequency located at the middle of two nearby Doppler filters. To always obtain reliable target detection in the phase-domain, we can use *Doppler-shifting* to move the arbitrary target Doppler to the middle of the interval by applying a shift frequency to the received RF signal. However, Doppler-shifting will also increase FFT leakage caused by the clutter and receiver noise. A detrending technique has been developed to reduce the leakage caused by clutter but still keep strong low frequency components caused by targets. Finally, we note that we have applied these algorithms to the Close In Weapon Support (CIWS) radar and showed that we could detect a –5 dB SCNR target in mainlobe clutter.

After detecting the target the next task will be to estimate the target direction. Target direction is measured by the elevation and azimuth angles, which are directly related to the phase differences between the antenna-quadrant spatial channels. Two methods, the direct method and the cross-correlation method, have been developed to estimate the detection angles. The first method uses the low frequency components directly measured from each individual channel. The measured low frequency component values obtained by PSD operation are scaled back by: $\varphi_i = C_d P_i^{0.5}$, where $i=1,2,3,4$; C_d is a scale constant; and P_i is the measured low frequency component values. The second approach obtains the measured low frequency component values by first conduct cross-correlation then PSD operations. There are six pairs of distinct cross-correlation functions between the four antenna channels: P_{12}, P_{13}, P_{14}, P_{23}, P_{24}, and P_{34}. Similar to the direct method discussed above, the scaling equation is $\varphi_{ij} = C_c P_{ij}^{0.5}$, where C_c is a scale constant. We have evaluated this approach using the previous simulated data. The estimated direction angles were ~10% for the first approach and ~3% for the second approach.

1.5 Spectral Analysis Codes

We propose to investigate advance code waveforms to suppress clutter for situations where standard statistical techniques become unstable. This section discusses a novel class of multiscale waveforms that possess a number of properties that are applicable to the KASSPER problem. By using a completely new approach to the classical theory of Walsh functions, we have developed a series of mathematical algorithms for the design of coding sequences – Spectral Analysis Codes (SAC) – that can be utilized specifically to detect and resolve spectral characteristics of target returns buried in clutter and noise. SAC design techniques can be employed both as (1) signal processing tools at the receiver as well as (2) in the generation of modulating sequences for pulse-coded waveforms.

1.5.1 Radar Processing *via* SACs

In order to separate the target from clutter return we are capable of producing a family of SAC codes with spectral characteristics which can be customized respond "flatly" to the kind of clutter return determined by the application. Once the SAC family is determined it can be used as a frequency analysis filter: we identify the target by tracing any fluctuations from the statistically expected value in the Power Spectral Density picture drawn by using our SAC family as a basis for the frequency transform. It is important to note that this "clutter-customized" power-spectrum estimation can be carried at several time-scales simultaneously. Furthermore, the approach suggested above is based on algorithms whose complexity does not exceed the one of classical Fourier-based peak-position estimation methods but with the additional advantage of being a more flexible scheme to adapt to different clutter/target characteristics.

1.5.2 SAC-modulated Radar Waveforms

SAC families of coding sequences can be modeled to suitably comply with a variety of time-frequency analysis requirements. Both their Frequency response and Power Spectral Density can be designed rather easily to be close to AWGN or highly coherent, depending on the requirements imposed by the application. In addition to that, the theoretical approach developed allows for the design of SAC code families that exhibit a prescribed auto-and cross- correlation pattern. This is a valuable characteristic, enabling the customization of coded waveforms to take advantage of the specific performance of the transmitter/receiver. The characteristics of the auto-correlation path of the pulse-compressed signals are adjustable, *e.g.*, to the specific constraints dictated by the antenna pattern under consideration. Furthermore, our techniques can be implemented in a scenario where our target is illuminated by two or more radar signals in order to optimize the cross-correlation performance. Another remarkable property of the new coded waveforms is their potential to be operated at different scales whenever the need arises to provide multiple resolution modes, *e.g.*, in a ranging application. The availability of these coded waveforms affords the possibility of improved clutter suppression.

1.5.3 Mathematical Strategy for SACs

We first consider the recursive formula defining Walsh functions:

$$\begin{aligned} W_0(x) &= 1 \\ W_{2n}(x) &= W_n(2x) + W_n(2x-1) \\ W_{2n+1}(x) &= W_n(2x) - W_n(2x-1) \end{aligned}$$

And observe that the rule allowing to move from one scale to the next is in fact just one out of the many possible unitary transformations that can be used to produce a family of orthogonal functions with the same time-frequency characteristics at each scale. In a more general approach we investigate a series of multi-scale transformations giving rise – by means of the very same iterative Walsh scheme – to a whole class of new codes that exhibit the same auto- and cross-correlation characteristics at each scale.
The modified scheme can be described as:

$$\begin{aligned} C_0(x) &= \underline{v} \\ C_{2n}(x) &= S_1(C_n(2x)) + T_1(C_n(2x-1)) \\ C_{2n+1}(x) &= S_2(C_n(2x)) - T_2(C_n(2x-1)) \end{aligned}$$

Where S's and T's are suitably "well-behaved" transformations and v is the initial vector possessing the desired characteristics. In this context we chose S and T among those transformations which will preserve the auto-correlation pattern. An example of this procedure is given by the so-called Rudin-Shapiro sequence:

$$\begin{aligned} C_0(x) &= 1 \\ C_{2n}(x) &= (1-i)C_n(2x) + (1+i)C_n(2x-1) \\ C_{2n+1}(x) &= (1+i)C_n(2x) - (1-i)C_n(2x-1) \end{aligned}$$

It should be noted here that in the case of Walsh functions and Rudin-Shapiro sequences the transformations S and T are multiplications by a (real or complex) number of modulus one. This is not at all the only possible choice. The "good" choices for S and T can be efficiently described by making use of tools arising from Harmonic Analysis, so that the emphasis can be set on the space characteristics (auto- and cross-correlation, number of phases, *etc.*) or the frequency content of the resulting coded signals. The complexity of these algorithms is going to be directly proportional to N Log(N) times the complexity of the transformations S and T. The described procedure is illustrated in Figure 17: the transformations S and T are "correlation-preserving" mappings, while the initial auto-correlation pattern is designed by computer.

The case where we want to model our multi-scale SAC codes to have a pre-assigned frequency content is entirely similar. Our SAC family may for example be a DFT-like set with a fixed number of phases. By convolving the sampled return of a Radar receiver with an ad-hoc SAC sequence we can spot fluctuations in the Power Spectral Density of the signal, possibly due to the presence of a target.

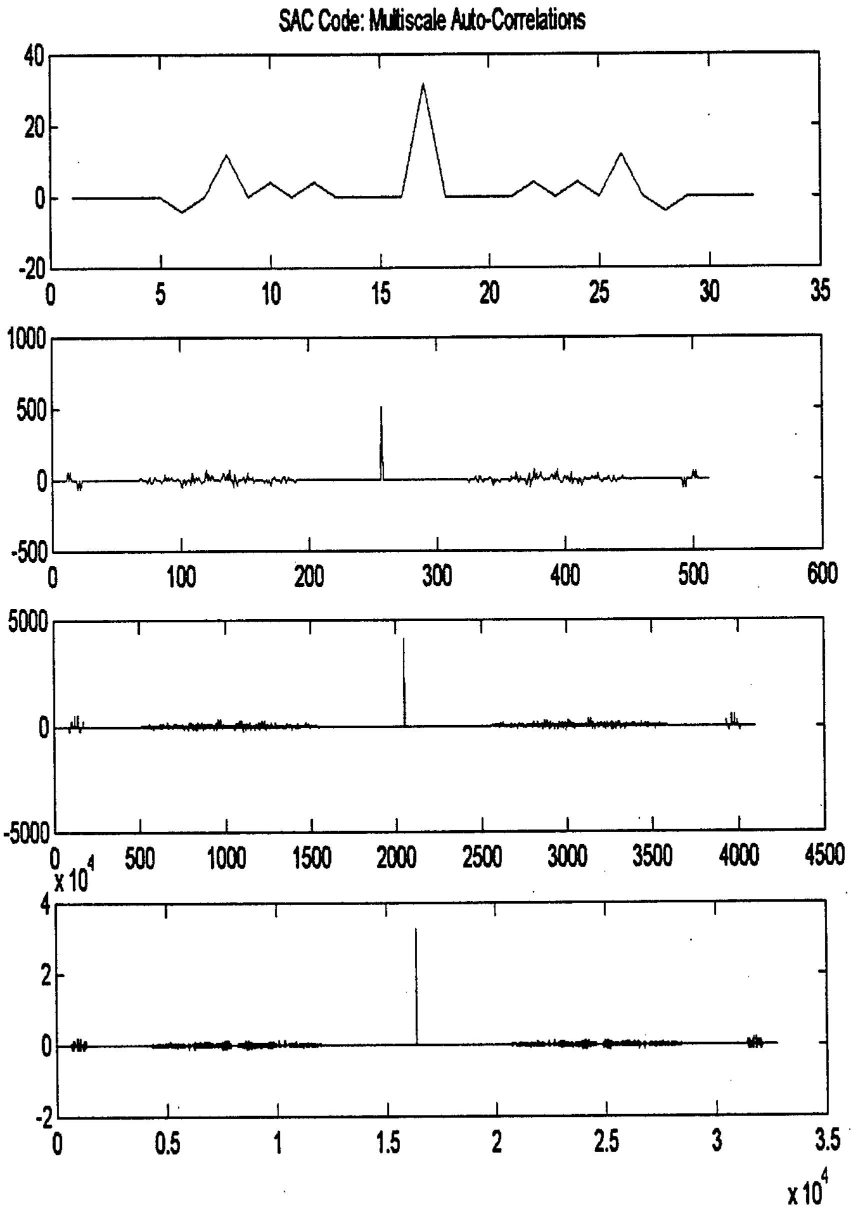


Figure 17: Illustration of SAC Correlation Properties

1.6 Multiresolution STAP

One line of investigation that appeared to have great initial promise was AAP in wavelet domain. We ultimately did not pursue this research in any significant way because of the overhead needed for the relatively small matrix sizes involved; this held true even for anticipated missile systems. Nonetheless, it is worth summarizing some of the more significant results. We concentrated on compressing the covariance matrix for broadband jamming. We selected an m-band wavelet filterbank, since this had been shown in the literature to lead to a banded (sparse) structure in the compressed domain. This is illustrated in Figure 18.

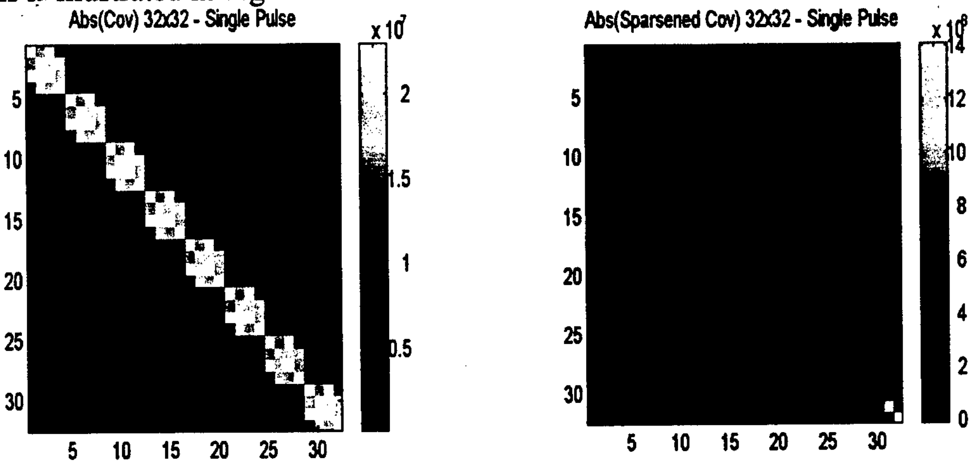


Figure 18: Wavelet Transform of TSI Covariance Matrix

We have also found that ~90% of the wavelet coefficients could be discarded before there is any significant impact on the adaptive processing performance. We were also able to verify a very interesting result that had been reported in the literature – that for some steering vectors performance was actually enhanced in the compressed domain. These results are shown in Figure 19. While the performance improvement was significant, we were never able to achieve it in the robust and repeatable manner necessary to implement it in flight code.

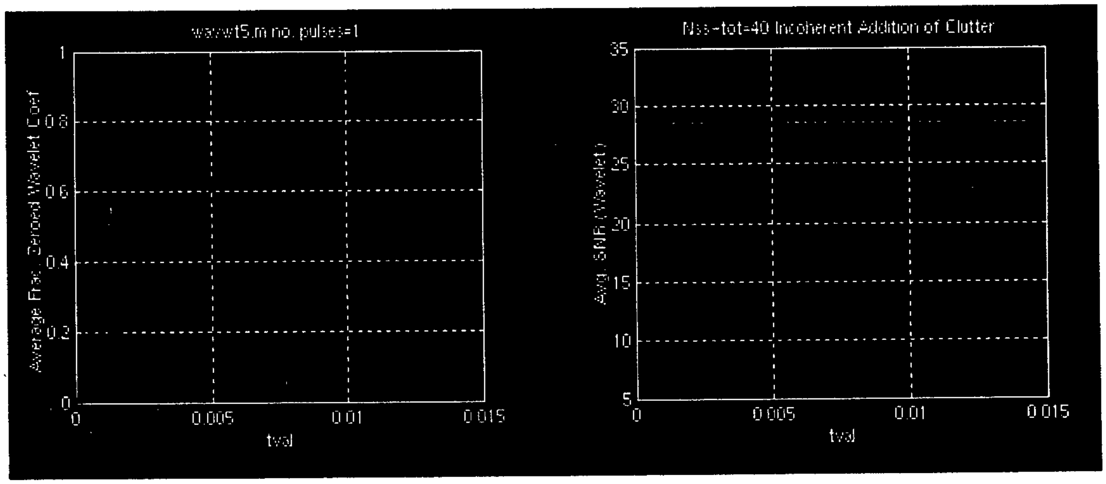


Figure 19: (a) SNR versus Compression; (b) Compression can Enhance SNR

2.0 Detection, Classification and Restoration for UCIR Imagery

Infrared imaging sensors that operate without cryogenic cooling have the potential to provide the military user with exceptional night vision capabilities packaged in a device of extremely small size, weight, and power. This would significantly reduce the cost and accelerate the implementation of sensors for applications such as targeting, surveillance, and threat warning. However, the performance of UCIR sensors is still inferior to that of cooled sensors. This performance gap limits the number of applications and precludes the widespread use of UCIR sensors in military missions. While UCIR sensors are being considered as replacements for cooled sensors in some applications, perhaps more importantly, the unique characteristics of the uncooled sensors are spawning novel uses of the technology. Very small, low-power sensors with moderate levels of performance are possible with the UCIR technology. In addition, applications such as micro-vehicles and robotics demand an extremely lightweight imaging sensor.

There is currently a great deal of interest in UCIR sensors within the Department of Defense community for Automatic Target Acquisition (ATA) for smart munitions. The obvious attraction of UCIR sensors over the more traditional cooled sensors is their low cost. This cost advantage should become even more pronounced with the economies-of-scale expected from commercial applications, particularly the automotive industry. The trade-off for achieving this greatly reduced cost is degradation in image quality that places a significantly greater burden on the ATA algorithms. There are a number of very exciting new approaches, such as multiplexed imaging, currently being investigated to improve the performance of UCIR sensors; however, they are not mature enough to be implemented in current generations of munitions.

2.1 Description of Imagery

For the processing described in the current paper, we use Raytheon's modeling environment – simulation technology image generation (STIG) – that was specifically developed for uncooled IR scene generation. STIG is a high fidelity modeling tool that uses a collection of measured backgrounds, a series of turntable target data, a detailed model of the IR sensor and a target facet model to generate realistic scenes of targets in clutter. This simulation allows the target-to-background contrast to be adjusted and permits the inclusion of a variety of discrete clutter types. A flow diagram for the STIG simulation is shown in Figure 20 below.

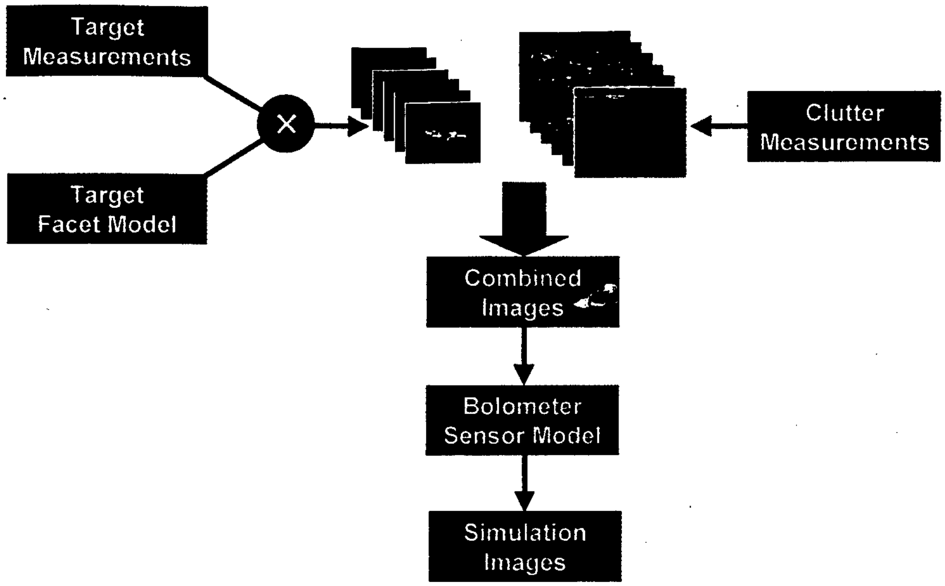


Figure 20: Simulated UCIR Image Generation

In addition to the simulated imagery, we also had available for analysis data from two CFTs. One CFT was conducted in Yuma, Arizona. For this test, the clutter was dominated by large discretes – here ocotillo shrubs. The second CFT was conducted in Huntsville, Alabama. For this test, the clutter was more benign. Figure 21 shows some examples of uncooled IR imagery from both simulation and CFT, under a variety of conditions. Clearly, this imagery presents a very challenging ATA problem. In the next section, we discuss some of the processing results in more detail.


Simulation - Yuma Clutter/Morning/1000m

Simulation - Yuma Clutter/Afternoon/1000m


Captive Flight Data - Yuma

Captive Flight Data - Huntsville

Figure 21: Example UCIR Imagery from Simulation and Flight

2.2 Local Singular Value Decomposition (LSVD) Algorithm

The Local Singular Value Decomposition (LSVD) is evaluated as a detection algorithm for uncooled infrared (UCIR) imagery. LSVD uses local statistics to identify anomalous regions and is very good at identifying local texture differences. The LSVD appears to work quite well on UCIR imagery, which tends to be extremely low contrast and blurry. Results are presented for both simulation and captive flight test (CFT) data.

2.2.1 Algorithm Description

The LSVD algorithm was developed for background estimation and anomaly detection. The LSVD algorithm is based on local statistical analyses and is a two-pass algorithm. It is based on the concept that each pixel value in an image can be expanded into a basis that consists of the surrounding pixels with coefficients corresponding to their values. This is shown schematically in Figure 22 for a pixel labeled by its eight nearest neighbors. In this way, it is possible to construct a distance metric from one pixel to another and, in particular, it is possible to introduce a distance from each pixel to a selected background region. Those pixels that are above a certain distance from the background are labeled as anomalies. Moreover, by selecting a background region, it is possible to estimate its local statistics and, in particular, to construct a set of singular values and right/left singular vectors. The right singular vectors can be used to 'rotate' each pixel in the image into the background coordinate system. The hoped for advantage of using this new coordinate system is that there are only a very few significant singular values, thus lowering the complexity for isolating and identifying pixels that are significantly different from the local background.

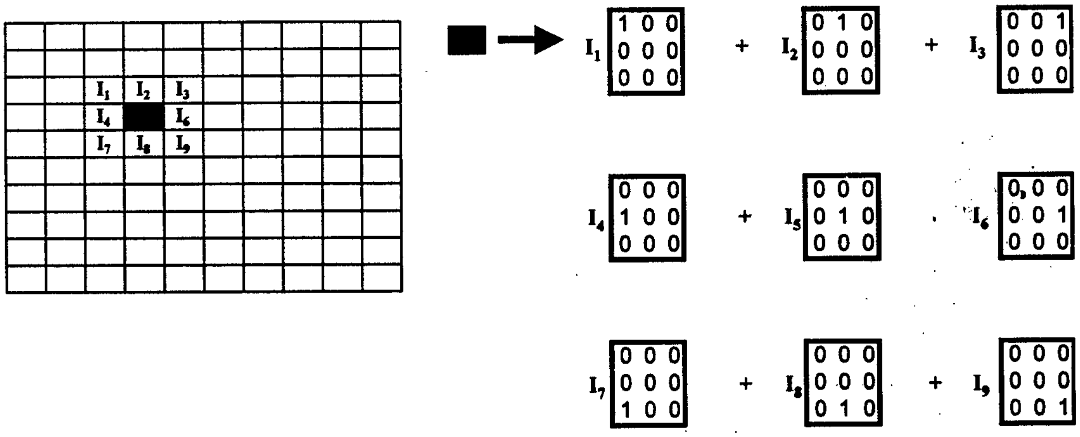


Figure 22: Sample expansion of a given pixel

- The LSVD algorithm is summarized below and illustrated in Figure 23:
  - "Background" region in image or set of images chosen
  - Principal components (PC) found for set of all *NxN* blocks of pixels in "background"
  - Coordinates found for each pixel via canonical *NxN* block
  - Distance to average in PC system determines anomalies:
  - Image *F* with pixels *p* expressed as B' $\cap$ A'
    - $A' = \{p \mid d(p) < s^2\}$

  - $B' = \{p \mid d(p) \geq s^2\}$
- Process iterated to give "second anomalies"

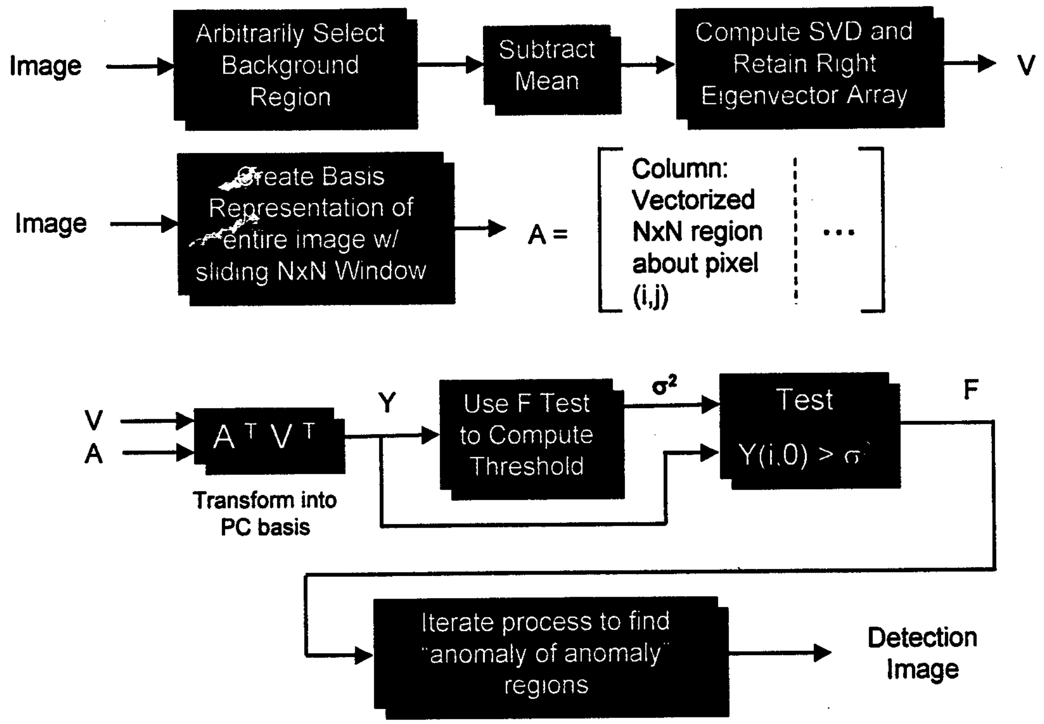


Figure 23: Flow Diagram for LSVD

A sample output of the LSVD algorithm is shown in Figure 24.

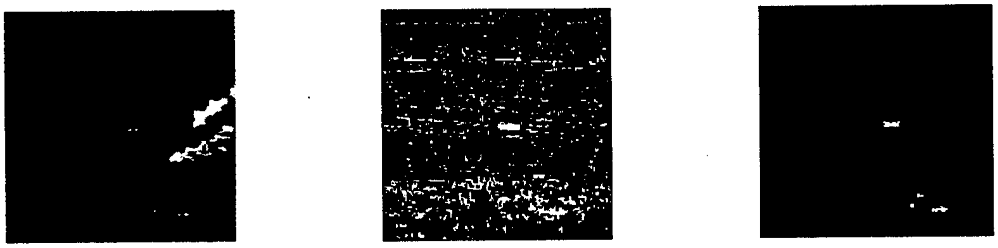


Figure 24: (a) Original Image; (b) First Anomalies; (c) Second Anomalies

2.2.2 Results

The STIG was used to generate a set of imagery with a clutter background collected from Yuma Proving Grounds. The data sets consisted of two different subsets from the morning and afternoon. For each of these cases, we generated images at two different acquisition ranges – 1000 m and 1500 m – and three different aspect angles – 0°, 150 ° and 300 °. Within these image sets, the target-to-background contrast was selected randomly. There were also two slightly different clutter types used, although they both correspond to desert clutter. The results are presented in Table 1 for clutter type 1 and in Table 2 for clutter type 2. Table 3 presents the processing results for data collected from the recent Yuma CFT.

We need to say a few words about how the results were calculated. First our detection algorithm was run. For the results presented below, the detection algorithm was the LSVD but we have also investigated a number of other algorithms. The output of the LSVD algorithm is a series of anomalous regions. For each detected region, we calculated its centroid and checked to see if it fell within the target truth region. Multiple centroids on the target are only counted once. The centroids that did not fall within the target truth region were counted as false alarms. The total probability of detection was found by calculating the total number of detections divided by the total number of images. Rather calculate a false alarm rate, the NetFires program requested that we report the false alarms as an average number of false alarms per image. This was done in the obvious was – the total number of false alarms was calculated for the image set and that number was divided by the total number of images.

	R = 600m Morning	R = 600m Noon	R = 1000m Morning	R = 1000m Noon
# Images Processed	126	126	126	126
# Targets Detected (P_D)	126 (100)	126 (100)	126 (1.00)	126 (1.00)
# False Alarms (FA/Image)	49 (.39)	265 (2.10)	11 (.09)	571 (4.53)
	R = 1500m Morning	R = 1500m Noon	R = 2000m Morning	R = 2000m Noon
# Images Processed	126	126	126	126
# Targets Detected (P_D)	125 (.992)	109 (.865)	121 (1.00)	108 (.85)
# False Alarms (FA/Image)	165 (.32)	1532 (12.16)	94 (.75)	1815 (14.40)

Table 1: Processing Results for Yuma 1 Simulation Data

	R = 600m Morning	R = 600m Noon	R = 1000m Morning	R = 1000m Noon
# Images Processed	60	60	60	60
# Targets Detected (P_D)	60 (1.0)	60 (1.0)	60 (1.0)	59 (.983)
# False Alarms (FA/Image)	3 (.05)	489 (8.11)	2 (.03)	452 (7.53)
	R = 1500m Morning	R = 1500m Noon	R = 2000m Morning	R = 2000m Noon
# Images Processed	60	60	60	60
# Targets Detected (P_D)	60 (1.0)	50 (.833)	60 (100)	47 (.783)
# False Alarms (FA/Image)	7 (.12)	740 (12.33)	25 (.43)	2703 (45.1)

Table 2: Processing Results for Yuma 2 Simulation Data

We have also processed the imagery collected during the recent WSMR CFT. There were some difficulties with the Non-uniformity Compensation (NUC) algorithm as well as some timing difficulties and as a result only a limited amount of image data was available for processing. The results are presented in Table 3. An example Receiver

Operating Characteristic (ROC) curve for the WSMR CFT at 1500m is shown in Figure 25. Because there are a very large number of parameters that can be varied for the LSVD algorithm, we restricted ourselves to the confidence limit for outlier determination and the tolerance level used to control convergence for the iteration to determine the second set of anomalies. Since there are a relatively small number of data points, the ROC curve tends to be somewhat 'jumpy.'

	R = 600m	R = 1000m	R = 1500m
# Images Processed	16	32	31
# Targets Detected (P_D)	16 (1.0)	31 (0.968)	27 (.87)
# False Alarms (FA/Image)	51 (3.19)	314 (9.81)	504 (16.26)

Table 3: Processing Results for WSMR CFT Data

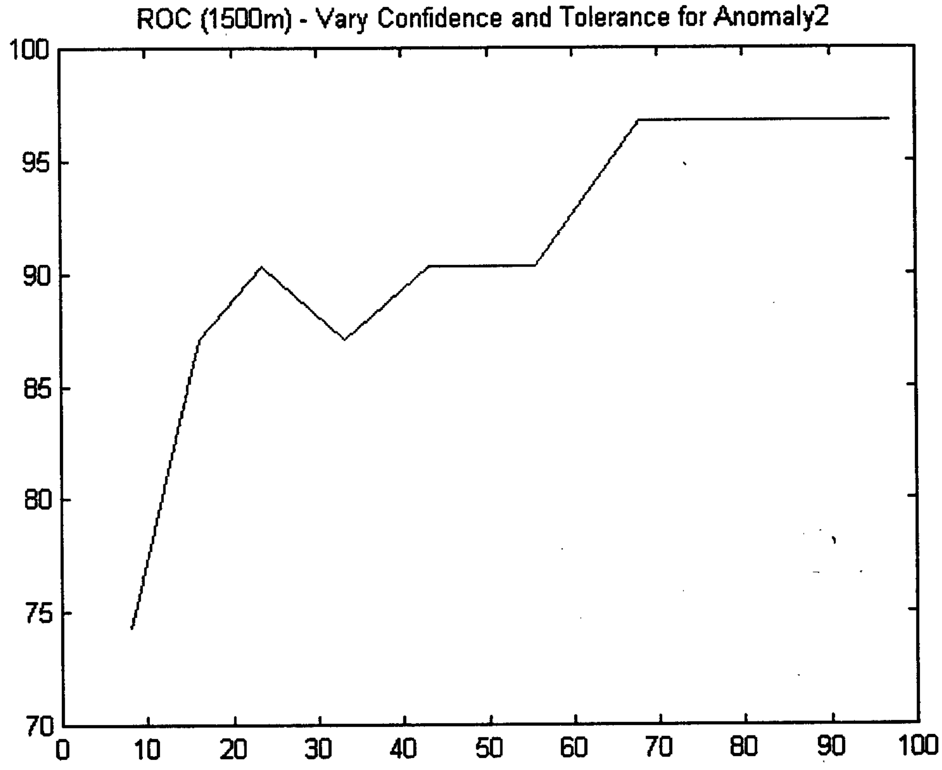


Figure 25: Example ROC for CFT Data Set

2.2.3 LSVD Post-Processing Using the Borrowed Strength Algorithm

We have been investigating a number of algorithms to post-process the anomalous regions identified by the LSVD algorithms. One approach in particular, the *Borrowed Strength Algorithm* (BSA) (described in section 2.7), shows a great deal of promise. It is used to significantly reduce the number of anomalies that the LSVD algorithm generates. The following image processing example illustrates this point.

Figure 26 shows an original image (a), together with its "textured" version (b). The textured image is computed as the local σ/μ value for each pixel's 3 by 3 neighborhood in the original image. (This neighborhood corresponds to choosing "s = 1" in section 2.7; "s = 2" averages a 5 by 5 region.) The computed LSVD anomalies obtained from processing the image in Figure 26(a) are shown in Figure 27(a), a binary image. Note that of the 240 by 320 total pixels in Figure 27(a), some 2,533 pixels are selected as possible anomalies.

Figure 26: (a) Example Target Image; (b) Textured Version

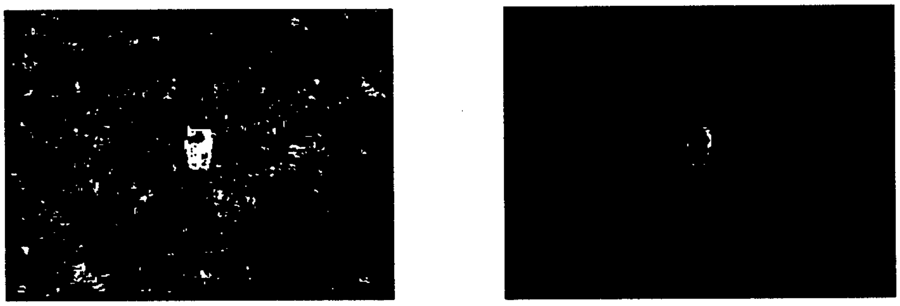


Figure 27: (a) LSVD Anomaly Results; (b) with BSA Processing

To augment the LSVD processing, the BSA works in the following way. First, a total of 2533 input values are chosen; they are the 2533 pixel values in the textured image 26(b) that correspond to the nonzero (white) anomalous pixels of Figure 27(a). Second, these data points are treated as belonging to a PDF (probability distribution function) and fitted to a sum of Gaussian normal functions, specified by their means, variances, and weighting coefficients. The number of Gaussian functions to be fit was allowed to vary between 1 and 10.

For this example case, the results of the various fits were quite robust. Although the best fit occurred when a total of 6 Gaussians were used, the "goodness of fit" was essentially unchanged when anywhere from 2 to 8 Gaussian normal functions were used to approximate the PDF of the 2533 data points.

The "winnowing" of the anomalous points (*i.e.*, moving from Figure 27(a) with 2533 points to Figure 27(b) with 55 points) comes from selecting only those pixels that are affiliated with the Gaussian normal function having the largest mean value. Following this "rule", the look of figure 27(b) remains quite unchanged when fits of 2, 3, 4, 5, or 6 normal functions are used to fit the full PDF; they would produce, respectively, 66, 48, 44, 48, and 55 lit pixels in the figure.

For this example, it is fortuitous that the special anomalous pixels can be separated so easily from the more common false alarms by merely selecting which Gaussian normal component they probably belong to. In more difficult cases, it is anticipated that the full machinery of the borrowed strength algorithm, *e.g.*, similarity matrix evaluations and integrated square error computations (see section 2.7), will be required. Nevertheless, it is very encouraging that essentially only the first, most basic, processing step of the BSA is able to so successfully treat the anomaly reduction problem associated with the LSVD algorithm.

2.2.4 Status of Program Transition and Algorithm Improvements

The most promising near-term insertion for UCIR sensor technology is the NetFires Precision Attack Munition (PAM) program. Other possibilities are Tank Extended Round Munition (TERM) and a classified program. The current schedule of flight tests is shown in Figure 28.

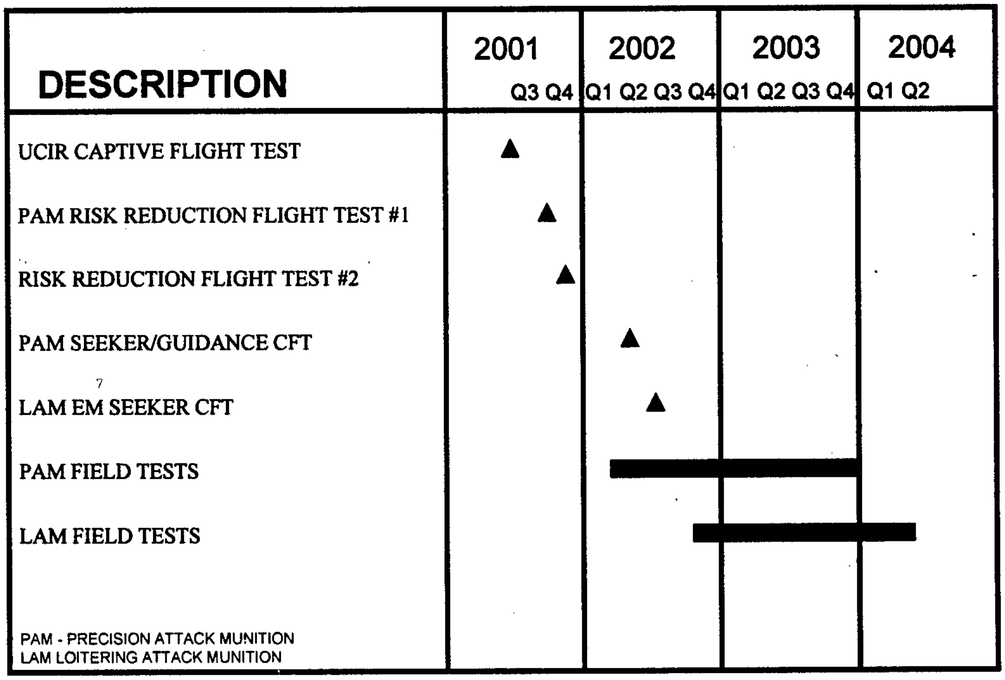


Figure 28: Road Map for NetFires LAM/PAM Programs

To make the next CFT for the NetFires PAM, the LSVD algorithm needs to be running near real time on flight hardware in the NetFires closed loop simulation by the middle of December 2001. That coincides with the end of our current program and is our final major goal for the program.

2.2.4.1 Program Transition Status

The MATLAB version of LSVD has been converted to C-Code and handed off to a real-time programming group in Dallas, Texas. The C-Code code has been ported as embedded code on the NetFires closed loop simulation. The embedded code has been run on the simulation and has gone through initial validation and verification.

Specific real time implementation steps performed to exploit current flight hardware architecture include:

- 2-D memory model converted to 1-D
- Optimized merge regions loop
- Eliminated redundant data movement
- Hard coded some array dimensions to allow loops to be unrolled
- Redesigned matrix multiplication routine to form subroutine capable of being implemented as AltiVec assembly code
- Redesigned outlier routine to form subroutine capable of being implemented as AltiVec assembly code

At this point, the processing times varied from 160-300 milliseconds. While this represented substantial progress, some final improvements were needed to reduce the overall run-time of the LSVD algorithm's implementation on the current flight hardware to the required 133 milliseconds. We note that all timing results are for a 350 MHz PowerPC G4 processor.

The run time of the second anomaly detection algorithm, anomaly2, is driven primarily by the number of pixels passed to it from the first anomaly detection algorithm, anomaly1. Because of this, the run time for anomaly2 is quite dynamic, a situation that is not acceptable for real time implementation. To address the problem of varying run times as well as to achieve the required overall run times, a routine was added to histogram the distance values generated in anomaly1. Two approaches were then investigated. The first approach limited the number of pixels passed by anomaly1 detection algorithm to anomaly2 such that the overall timing requirements were consistently met. The second approach eliminated the anomaly2 algorithm entirely by further limiting number of pixels kept from the out put of anomaly1. Since there were up to four processors available, tests were conducted of breaking the image up into a number of subimages and process each in parallel. It was found that approximately a 20-pixel overlap was required. The latest processing results are presented in Figure 29.

SVD Run Log						
SVD Source	V4 Main	V4 Main	V4 Main	V4 Main	V4 Main	V4 Main
Range (meters)	1000	1000	1000	1000	1000	1000
Region low/upper limits	8/600	8/600	8/600	8/600	8/600	12/500
Image partitioning	Whole	Whole	Whole	Whole	Whole	Whole
Image Normalization	New	New	New	New	New	New
Matmul/Outlier Implementation	Matmul/Outlier	Matmul/Outlier	Matmul/Outlier	MatMulDistMap	MatMulDistMap	MatMulDistMap
Matrix Multiply Implementation	C	C	C	C	AltiVec Asm	AltiVec Asm
Anomoly1 Output Pixel Limit	Unlimited	1000	400	250	250	250
Anomoly2 Output Pixel Limit	Unlimited	Unlimited	100	na	na	na
Anomoly2 processing	Forced	Forced	NOT Forced	NOT Run	NOT Run	NOT Run
Closing processing	Enabled	Enabled	Enabled	Enabled	Disabled	Disabled
Num images processed	120	120	120	120	120	120
Num targets detected	113	111	114	116	117	116
Additional sub-targets detected	35	35	35	7	40	30
Num False Alarms (FA)	313	39	313	437	318	168
Probability of Detection (PD)	94.2%	92.5%	95.0%	96.7%	97.5%	96.7%
FA/image	2.61	0.325	2.61	3.64	2.65	1.4
FA Rate	0.000452	0.000057	0.000452	0.000637	0.000463	0.000245
Mean area	126.212	82.72	126.149	229.164	166.837	167.379
Sigma	78.43	47.15	78.08	116.5	86.59	87.24
Total Execution Time (mSec)*	187 to 350	185 to 215	173 to 193	154 to 157	125	125
Staged Execution Time**	Total / 2	Total / 2	Total / 2	Total / 2	Total / 2	Total / 2

** All execution times are in milli-seconds (mSec) and represent SVD running on a single 350 MHz PowerPC G4 processor*

*** Staged execution time is generally equal to the total divided by 2.*

Figure 29: Real Time Processing Results Summary

So, in summary, we expect to have the LSVD algorithm available for transition to the NetFires program for their next CFT. Currently, the LSVD algorithm has the best performance of any of the detection algorithms NetFires has evaluated. The performance advantage is particularly stunning for the 1500m and 2000m afternoon data. – the LSVD algorithm is out performing the next best algorithm by about 45 percentage points in detection (83% to 38% at 1500m and 78% to 32% at 2000m) with comparable false alarms per image. We are working closely with the NetFires Program Office to make this transition a reality.

2.2.4.2 Possible LSVD Improvement

This section describes improvements made to the LSVD algorithm developed by Professor Xiaobai Sun from Duke University. This work was not funded under the current contract and is derived from the LSVD code that is proprietary to FMAH. While we report these results here, the code is not considered a deliverable under the current contract. At the time the final report was being written, Professor Sun had made some significant improvements to the LSVD code. In particular, for the UCIR images provided, Professor Sun had achieved a 5x reduction in both memory requirement and arithmetic operations. Operational improvements such as singular value decomposition truncation by given numerical tolerance and an efficient update to enlarged neighbor size, are also being incorporated. These improvements will be quantified by processing the available image sets. We expect that this evaluation to be concluded before completion of the contract. We also believe that these improvements will have an even greater impact on target detection for SAR and hyperspectral imagery.

2.3 Anisotropic Image Diffusion

The last decade has seen the advent and development of a highly successful new image processing paradigm; it is variously known as anisotropic diffusion or nonlinear diffusion filtering. In this approach, particular types of parabolic partial differential equations are developed for processing noisy images. Generally, they are of the form:

$$\frac{\partial u}{\partial t} = div(D \cdot \nabla u),$$

where $u(x,t)$ represents the intensity level of pixel x in an image at pseudo-time t, D is the diffusion tensor, a positive definite symmetric matrix, and pixel x represents a location in the 2-dimensional image region. The initial condition is $u(\mathbf{x},0) = f(x)$, where $f(x)$ is the starting image, *i.e.*, the measured image that is to be sharpened or otherwise processed. Solutions of these PDEs (diffusive initial value problems) represent the evolution of raw images in "pseudo-time", as *across-edge* features are enhanced while, simultaneously, *along-edge* features are smoothed. During this process, reflective boundary conditions are applied along the image's boundary; these conditions act to guarantee a constant average intensity value over the whole image. The ultimate goal is to detect faint targets in noisy images.

In practice, these diffusion equations have two additional complicating, but necessary, factors: a regularization parameter σ and a contrast parameter λ. These help to, respectively, remove ill-posedness in the problem and enhance edges above a certain contrast level. A Gaussian smoothing function $K(x)$ is parameterized by σ:

$$K_\sigma(\mathbf{x}) = \frac{1}{2\pi\sigma^2} \cdot \exp\left(-\frac{\mathbf{x}\cdot\mathbf{x}}{2\sigma^2}\right)$$

The scalar diffusivity function g is parameterized by λ and is given by:

$$g\left(|\nabla f|^2\right) = \frac{1}{\sqrt{1 + |\nabla f|^2 / \lambda^2}}$$

The orthonormal system of eigenvectors v_1, v_2 of the diffusion tensor D is constructed such that v_1 is parallel to ∇u_σ and v_2 is perpendicular to ∇u_σ. In order to prefer smoothing along the edge to smoothing across it, one can choose the corresponding eigenvalues λ_1 and λ_2 (not to be confused with the contrast parameter λ) as $\lambda_2 = 1$ and

$$\lambda_1 = g(|\nabla u_\sigma|^2)$$

The smoothed intensity field, u_σ, is the convolution with the Gaussian smoothing function K:

$$u_\sigma(\mathbf{x}) = (K_\sigma * u)(\mathbf{x})$$

It should be emphasized that the diffusion equation shown here is only one example of using the anisotropic diffusion approach to solving image processing problems. The approach may be generalized in many ways, including using a sequence of real-time (not pseudo-time) images, using coincident multi-wavelength (colored) images or even adding another equation to perform temporal regularization:

$$\frac{dD}{dt} = \frac{1}{\tau}\left(F(\nabla u) - D\right),$$

where $\tau > 0$ is some delay parameter and $F(\nabla u)$ is basically a projection orthogonal to ∇u, if $|\nabla u|$ is larger than some contrast parameter λ.

2.3.1 Diffusion Filter Types

Several different types of finite difference stencils (filters) can approximate the solution of the diffusion equation. The filter types are defined by the structure of the diffusivity term in the diffusion equation and are conveniently broken into three classes

Linear Diffusion Filters: The diffusivity term D is a constant over the entire image. The subsequent processing smoothes the image in all directions equally. This form is equivalent to filtering the image with a Gaussian kernel. The filter does not discriminate in what is processed and will smooth away the noise and the targets equally.

Nonlinear Isotropic Diffusion Filters: The diffusivity term D is a scalar function of the magnitude of the local image gradient. This function is formed such that the image will be smoothed in relatively constant regions, and not smoothed in regions with large gradients that may represent object edges. Most of the literature on image diffusion processing is based on this type of filter. Unfortunately, this method does not perform well in eliminating impulse noise. The diffusion filter operates on a pixel by pixel basis and cannot take into account how this pixel differs from the local neighborhood.

Nonlinear Anisotropic Diffusion Filters: The diffusivity term D is a tensor. This tensor is a function of the differential structure of the evolving image. The filter derived from this formulation can smooth away the noise in the image without smoothing away the target. This filter does not suffer from the deficiencies of the isotropic formulation. In addition, the tensor formulation allows for a whole new paradigm, Coherence Diffusion Filtering (CDF). CDF formulates the problem based on *a priori* geometric information. Solutions can be created to the diffusion equation that are not based on preserving edges, but on enhancing the coherence of flow like textures.

2.3.2 Derivation of the Nonlinear Anisotropic Diffusion Filter

The remainder of this section will be devoted to the development and the application of the nonlinear anisotropic diffusion filters. Referring back to the diffusion equation, we can write this equation out in more detail:

$$\frac{\partial \mathrm{u}}{\partial \mathrm{t}} = \nabla \bullet (\mathrm{D}\nabla \mathrm{u}) \text{ where } D = \begin{bmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{bmatrix} \Rightarrow$$

$$\nabla \bullet \left(\begin{bmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{bmatrix} * \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix} \right) = \frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u}{\partial x} + b(x,y)\frac{\partial u}{\partial y} \right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial u}{\partial x} + c(x,y)\frac{\partial u}{\partial y} \right)$$

If the PDE is discretized using a symmetric scheme for the first order derivatives, one obtains a 5x5 stencil. This can be reduced to a 3 x 3 stencil by allowing asymmetric schemes. The first order derivatives are calculated by alternately approximating the inner and outer derivatives with a Forward Euler method (+) followed by a Backward Euler method (-). By averaging the two asymmetric components, one obtains an overall symmetric scheme.

$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u}{\partial x} \right) = \frac{1}{2}\left[\frac{\partial^{+}}{\partial x}\left(a(x,y)\frac{\partial^{-} u}{\partial x} \right) + \frac{\partial^{-}}{\partial x}\left(a(x,y)\frac{\partial^{+} u}{\partial x} \right) \right]$$

The mixed derivative terms in the PDE are approximated with central differences. These calculations can then be factored into like terms and the expression simplifies to:

$$\frac{du}{dt} = A(u)u$$

$$A(u) = \begin{bmatrix} -\frac{b_{i-1,j} + b_{i,j+1}}{4} & \frac{c_{i,j+1} + c_{i,j}}{2} & \frac{b_{i+1,j} + b_{i,j+1}}{4} \\ \frac{a_{i-1,j} + a_{i,j}}{2} & \frac{a_{i-1,j} + 2a_{i,j} + a_{i+1,j} + c_{i-1,j} + 2c_{i,j} + c_{i+1,j}}{2} & \frac{a_{i+1,j} + a_{i,j}}{2} \\ \frac{b_{i+1,j} + b_{i,j+1}}{4} & \frac{c_{i,j-1} + c_{i,j}}{2} & -\frac{b_{i+1,j} + b_{i,j-1}}{4} \end{bmatrix}.$$

The solution of the PDE is applied iteratively to the evolving image. For stability of the algorithm, the constant *dt* should always be less than 0.25.

$$U_{i,j}^{t+dt} = U_{i,j}^{t} + dt * \sum_{x=-1}^{1} \sum_{y=-1}^{1} a_{x,y} u_{i+x,j+y}$$

for every ij in the image and where $a_{0,0}$ corresponds to the center pixel of the stencil.

2.3.3 The Diffusion Tensor

In the diffusion literature, much of what is considered "filter design" consists of how you choose to construct the elements of the diffusion tensor *D*. In general, for the positive semi-definite matrix *D*, the eigenvectors should reflect the local edge orientation. The method used to calculate the eigenvalues is particular to the application. In order to have the eigenvectors of D reflect the local edge orientation, it is necessary to calculate a geometry tensor (J_o) for each pixel in the image.

$$Jo(\nabla u) = \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} & \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} & \frac{\partial^2 u}{\partial y^2} \end{bmatrix}$$

J_o has an orthonormal basis of eigenvectors which are parallel (v_1) and perpendicular (v_2) to the gradient. Unfortunately, this calculation can be skewed by noisy pixels, and a better estimate of the local orientation can be found by convolving J_o component wise with a Gaussian to obtain an average of the local orientation.

Most of the filters implemented in practice use the structure tensor J_ρ for a more robust local edge orientation estimate. The eigenvectors of J_ρ, are parallel (v_1) and perpendicular (v_2) to the averaged local edge orientation. The largest eigenvector of the tensor indicates the local orientation. The relation between the eigenvalues of J_ρ, μ_1 and μ_2, expresses the certainty of this statement. The larger μ_1 is compared to μ_2 (or vice versa), the more certain the direction of the local edge orientation.

$$D = \begin{bmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{bmatrix}$$

$$D = \begin{bmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{bmatrix} * \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} * \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$$

$$a = \lambda_1 {v_{11}}^2 + \lambda_2 {v_{21}}^2; \quad b = v_{11}v_{12}\lambda_1 + v_{21}v_{22}\lambda_2; \quad c = \lambda_1 {v_{12}}^2 + \lambda_2 {v_{22}}^2$$

2.3.4 Filter Design

The method you choose to calculate the eigenvalues for the diffusion tensor D determines the functionality of the diffusion algorithm. The first two methods shown below are a direct extension of the edge enhancing nonlinear isotropic methods. The main difference is that, for the isotropic case, the function used to choose how much you diffused in the direction perpendicular to the edge is a function of the magnitude of the local gradient. Below we have replaced this by making the g function dependent on μ_1, which is the first eigenvalue of your geometry tensor J_o or J_ρ. When using the non-averaged geometry tensor J_o, the first eigenvalue of J_o is the magnitude of the local gradient and the expressions for the isotropic and anisotropic case are identical. These first two functions for λ_1 are both monotonically decreasing functions, which range between 0 and 1. Simply, when λ_1 is 1, you will diffuse in the direction perpendicular to your edge, and when λ_1 is 0, you will not diffuse at all. The purpose of the function g is to allow the diffusion filter to not diffuse in the vicinity of targets, and to diffuse everywhere else. K is chosen to approximate the magnitude of the gradient that would exist due to the finite difference between a target to the background. The difference between the first two functions is obviously the rate of decay. The better your estimate of what K value is needed, the sharper you want the transition to be between $\lambda_1 = 1$ and $\lambda_1 = 0$.

#1

$$\lambda_1 = g(\mu_1)$$
$$\lambda_2 = 1$$
$$g(\mu_1) = \frac{1}{1 + \frac{\mu_1}{K^2}}$$

#2

$$\lambda_1 = g(\mu_1)$$
$$\lambda_2 = 1$$
$$g(\mu_1) = 1 - \exp\left(-C\left(\frac{\mu_1}{K}\right)^{-m}\right)$$

#3

$$\lambda_1 = \alpha$$
$$\lambda_2 = g\left((\mu_1 - \mu_2)^2\right)$$
$$g(s^2) = 1 - \exp\left(\frac{-s^2}{(K)^2}\right)$$

Filter design method number three can be used to enhance the coherence of flow like textures. This filter smoothes along the coherence direction v_2 with a diffusivity, λ_2, which increases with respect to the coherence ($\mu_1 - \mu_2$).

The initial hope for the diffusion algorithm was that it could be used for the detection stage of the ATA problem. Detection would be accomplished by successively applying the diffusion algorithm to the image to segment the background and the target. Figure 30 shows this process on UCIR imagery using filter method number 2. Generally it was found that the number of iterations required to perform this segmentation was extremely large, and the ability of the algorithm to perform this task was image dependent. In images where the largest contrast in the image was between the target and background, the segmentation could be performed. However, in the event that this was not the case, the segmentation would diffuse away the target. The diffused image shown in Figure 1 was processed through 200 iterations at 0.1 dt. The complete segmentation required near 20000 iterations at 0.1 dt. The computational cost of this algorithm, as well as the difficulty in determining the appropriate stopping point, appears to rule out the use if this algorithms for detection with UCIR imagery. We have begun some preliminary investigations into using anisotropic image diffusion for other sensor modalities, such as SAR and LADAR. We next discuss possible applications related to pre- and post-processing of UCIR imagery.

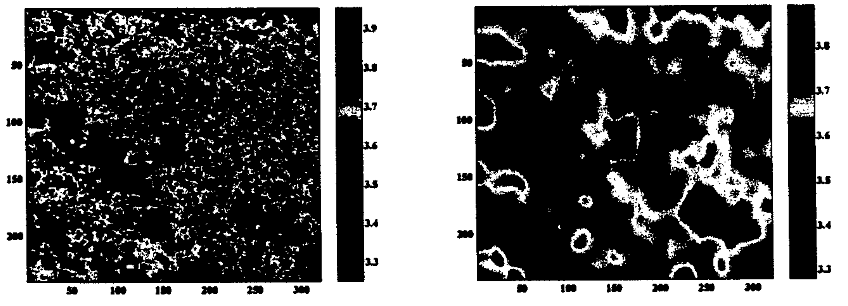


Figure 30: (a) Original Image (b) Diffused Image

Consider the use of the diffusion algorithm for image enhancement and noise removal. Figure 31 shows and an example of a distorted UCIR image that was enhanced using filter number 2. The original image was distorted with Gaussian noise and recovered using three iterations of the diffusion algorithm at 0.1 dt. Similar results were obtained using filter number 1.

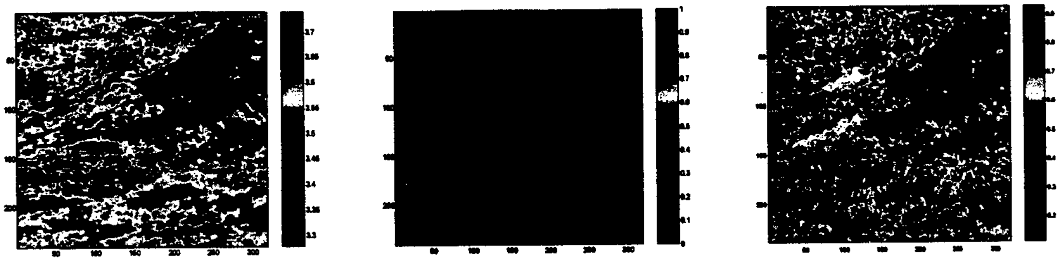


Figure 31: (a) Original Image; (b) Noisy Image; (c) Recovered Image

Filter design method number three can be used to enhance the coherence of flow like textures. This filter smoothes along the coherence direction v_2 with a diffusivity, λ_2, that increases with respect to the coherence, $(\mu_1 - \mu_2)$. Here μ_1, μ_2 are the eigenvalues of the averaged geometry tensor. We have not as yet identified a compelling application of this filter to UCIR imagery. Perhaps one of the most intriguing aspects of the anisotropic image diffusion approach is the ability to incorporate external information into the potential term. This is very much in the spirit of the new DARPA DSO Integrated Sensing and Processing (ISP) program.

2.4 Template matching using Radon/Fourier transforms and LSDB

This section describes the first of two small investigative studies that were funded under this contract. These were not included in the original proposal but arose out of some very recent research published by students of Professor Gregory Beylkin and Professor Ronald Coifman. The objective of this first study was to extract robust object features from images for recognition and template matching using best basis methods. Professor Naoki Saito of the University of California, Davis is leading this research effort. The objects of interest are imaged with different rotations, dilations, and translations (*c.f.,* Figure 32) as well as local variations (*e.g.,* shadows or other non-stationary noise). Initial research focused on rotations and dilations. The main idea is to compute the rotation and dilation invariant representation and then to characterize the features by the least statistically dependent basis (LSDB).

Suppose all the objects are centered but rotated and isotropically dilated. Then, a slice of the Radon transform gives a 1D function of the rotated angle. The key observation is that rotation and isotropic dilation of the object are equivalent to translation and amplitude scaling of this 1D curve. Therefore, the normalized magnitude of the Fourier transform of this 1D curve gives us the invariant signature of the object with respect to rotations and dilations. The inverse Fourier transform is applied to this curve to get spatial features invariant to rotations and dilations, which are more intuitive than the frequency features. The curves obtained from several rotated versions of the same objects are now similar in appearance and the only differences are due to sensor noise and other local variations. This is illustrated in Figure 33.

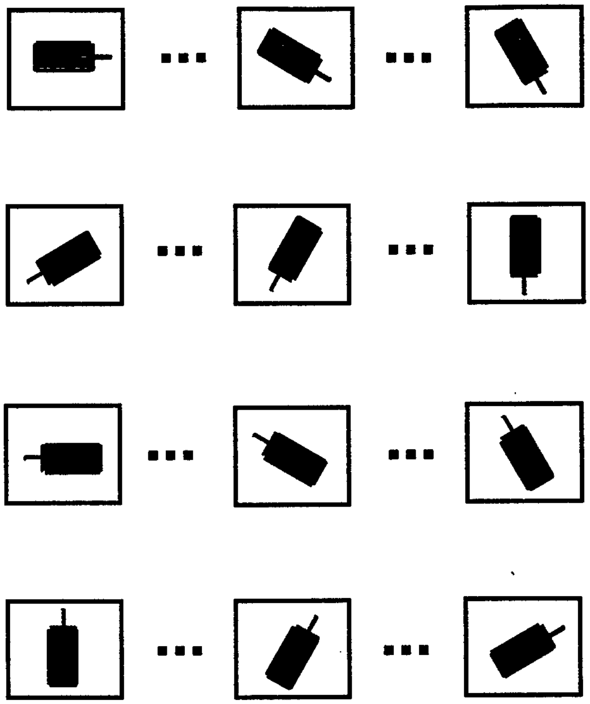
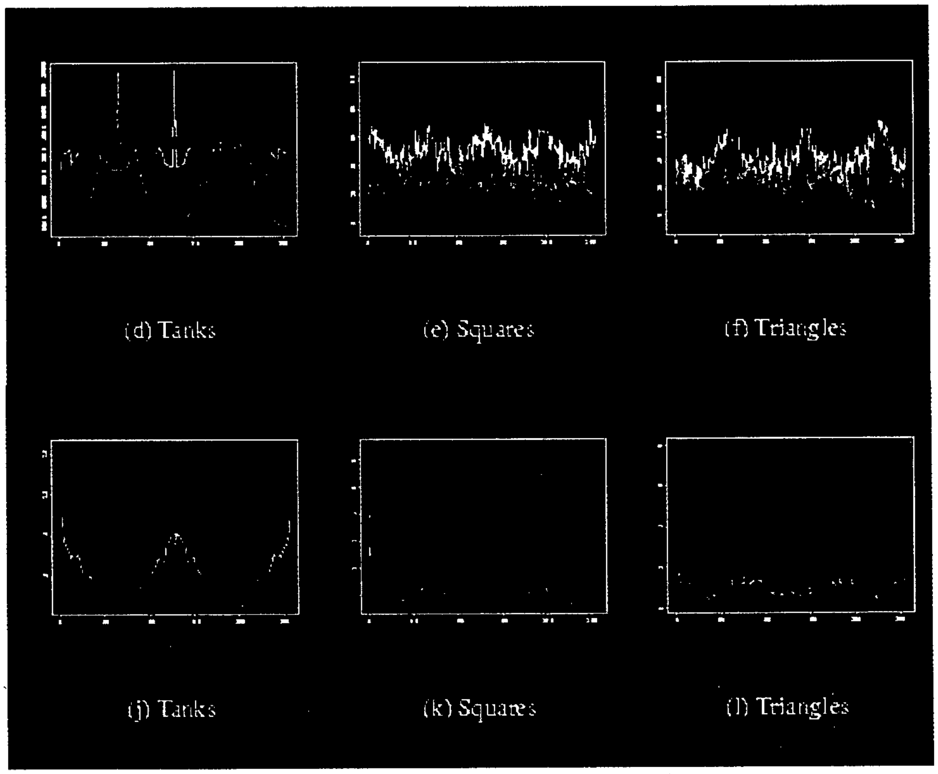


Figure 32: 2-D Image of Tank at different Aspect Angles



(d) Tanks (e) Squares (f) Triangles

(j) Tanks (k) Squares (l) Triangles

Figure 33: Removal of Rotation and Dilation Effects

Finally, the localized basis functions are computed using the LSDB algorithm and are then used to find features characterizing this object that are close to being statistically independent. The top 30 LSBD vectors are shown in Figure 34.

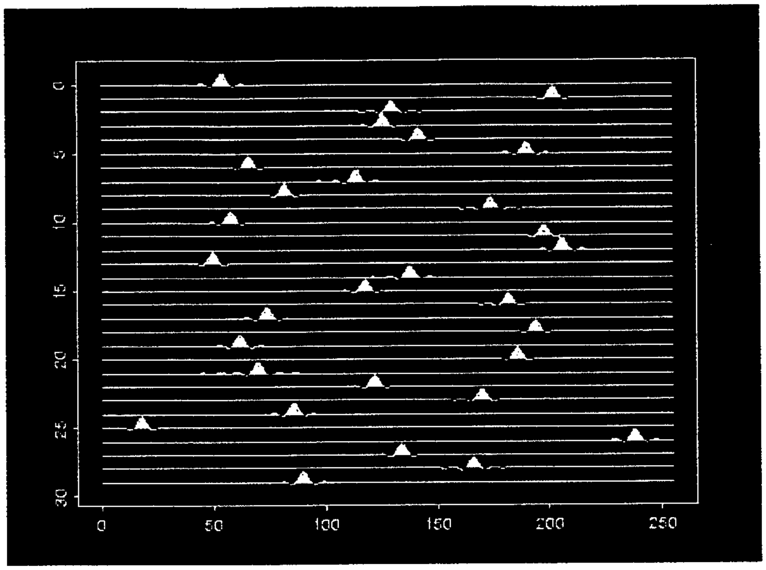


Figure 34: Top 30 LSDB Feature Vectors

2.5 Multiresolution Anisotropic Image Diffusion (MAID)

This research topic was motivated by the results of processing LADAR imagery with the LSVD algorithm after the imagery had been corrected for pixel dropouts. The difficulty arises because the LSVD algorithm is looking for differences in local statistics and most standard approaches to replacing pixel dropouts results in changes to the local statistics. The MAID algorithm was developed by FMAH and is based on the application of a diffusion filter and smoothing conjugate-gradient iteration recursively on multiple scales. The algorithm is fast and robust. A flow diagram for the MAID algorithm is given in Figure 35.

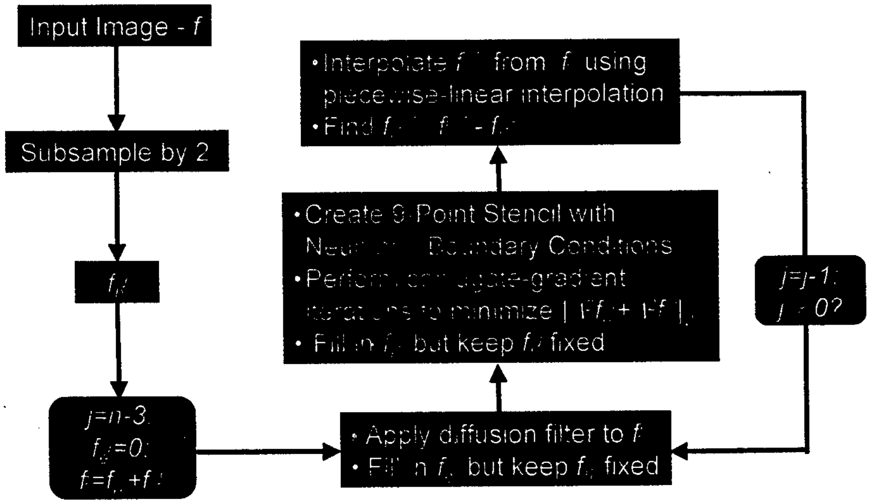


Figure 35: MAID Algorithm Flow Chart

Because of the sensitivity of distributing LADAR flight data, a mask was created from some especially badly corrupted flight data[1]. A simulated range image was created with IRMA and the mask was used to reproduce the pixel dropouts. The initial performance of the MAID algorithm can be evaluated visually in Figure 36.

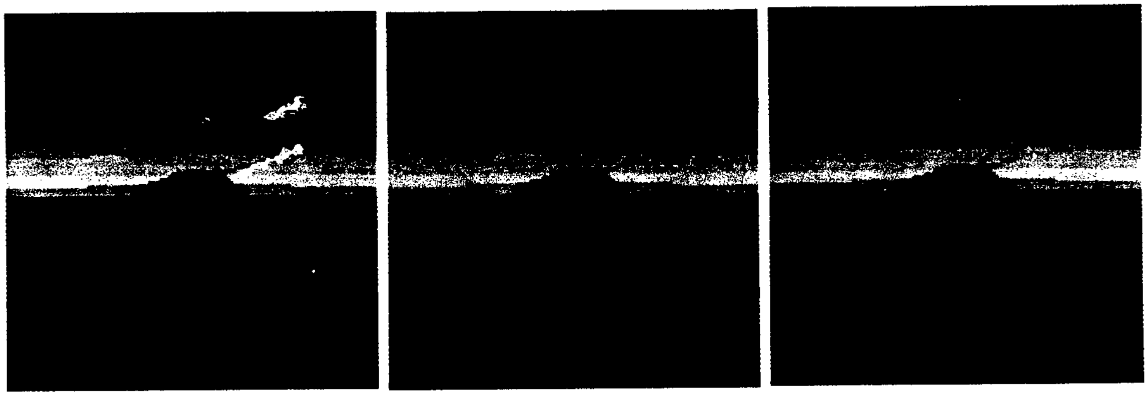


Figure 36: (a) IRMA Image; (b) Corrupted Image; (c) MAID Processed Image

It is useful to study the behavior of the wavelet coefficients as the MAID algorithm progresses through it iterations. The results are shown in Figure 37, where the reduction in number of significant coefficients shows quantitative improvement in the data.

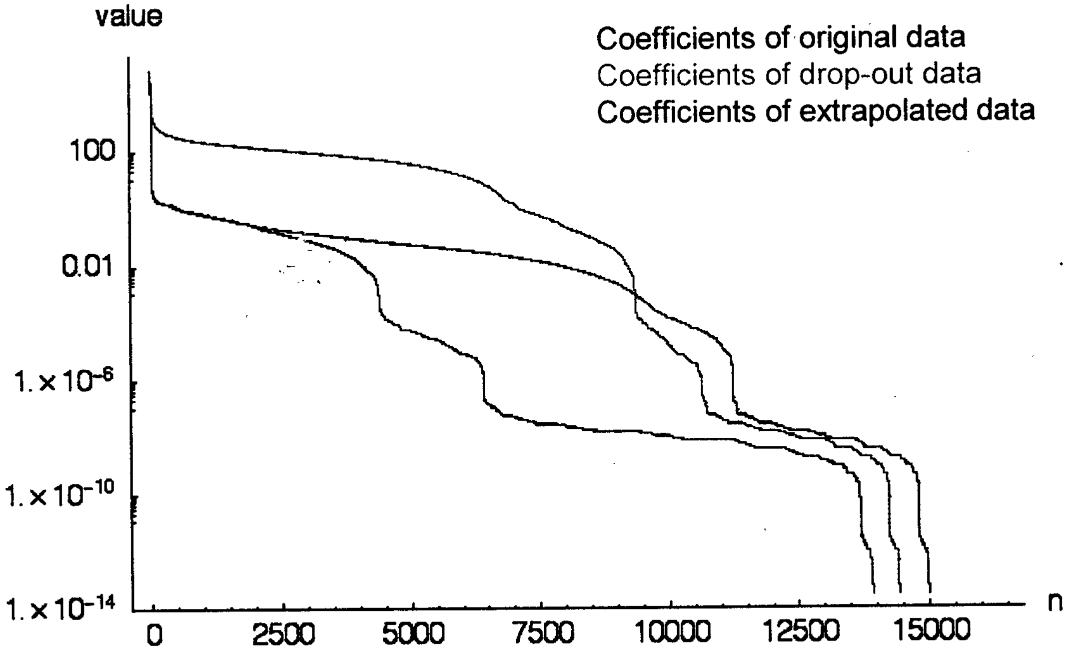


Figure 37: Comparison of Wavelet Coefficients

---

[1] This data was obtained very early on in the flight program before the final flight hardware was available. The severe dropouts were not present in the CFT.

2.6 Multiresolution Image Processing

Multiresolution image processing via the discrete wavelet transform (DWT) has been successfully applied to the areas of image compression, image denoising, and edge detection. The result of applying an m-level 2-D DWT to an image is a set of subimages LLp, LHp, HLp and HHp, where p=1,2,.., m. The LHp, HLp and HHp emphasize vertical, horizontal and diagonal edges respectively. The area of multiscale image processing is quite broad. What we report here are some of the results for two of the more promising algorithms for UCIR imagery: target detection and edge detection. For our analysis, we have used Daubechies and spline wavelets.

2.6.1 Wavelet Subband Product (WASP) algorithm for Target Detection

Subband products are evaluated as a detection algorithm for UCIR imagery. This algorithm uses the products of particular subbands of the 2-D DWT identify ROIs. We considered one and two level decompositions using the Daubechies and spline wavelet transforms. This algorithm works reasonably well on UCIR imagery, although its performance is clear inferior to the LSVD algorithm. This is an extremely efficient algorithm, however, and so might be considered for fusing with the LSVD. The best results were obtained using a two level decomposition with spline wavelets. In fact, this is the only instance that we have obtained where the choice of wavelet family made a significant difference. Results are presented for both simulation and CFT data. The WASP algorithm, which exploits the inherent rectangular structure of man made objects or vehicles and uses wavelet subbands to enhance horizontal and vertical edge information, is summarized below and illustrated in Figure 38

- Reflect N x M input image to K x K image, where $K = 2^q$
- Wavelet decompose image and extract N/2 x M/2 regions from LH1 and HL1
- Perform element-by-element multiplication, SubProd1, of LH1 and HL1
- Wavelet decompose image and extract N/4 x M/4 regions from LH2 and HL2
- Perform element-by-element multiplication, SubProd1, of LH2 and HL2
- Threshold SubProd1 and SubProd2 to obtain detection maps

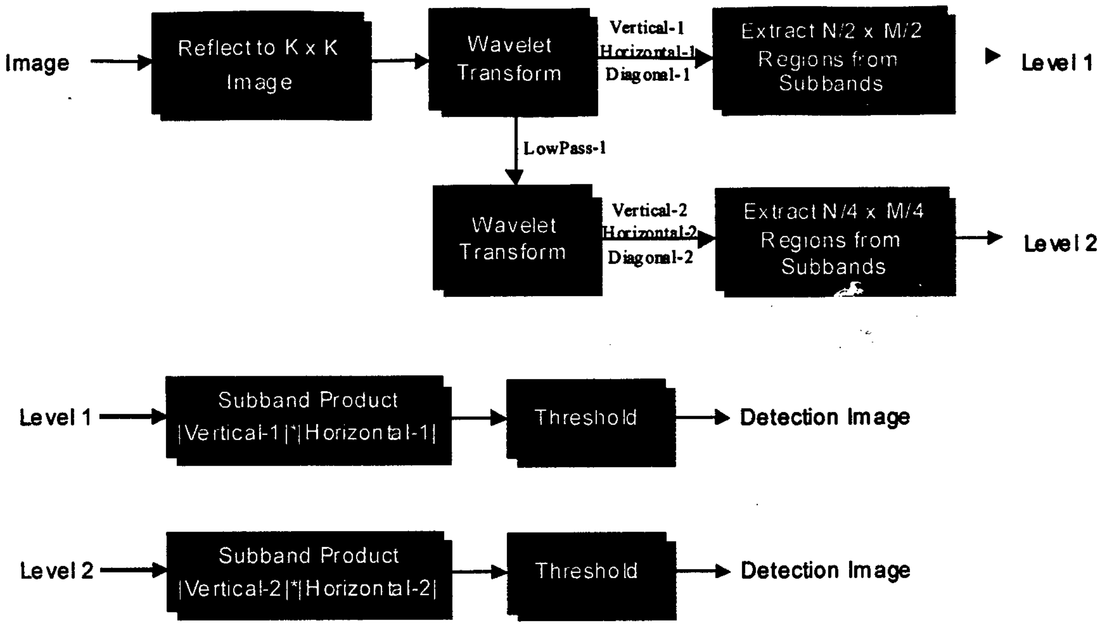


Figure 38: Flow Diagram for Wavelet Subband Product

Sample outputs of the WASP are shown in next four figures. Figure 39 and Figure 40 illustrate detection results for one-level and two-level decompositions using spline wavelets. Figure 41 and Figure 42 illustrate detection results for one-level and two-level decompositions using Daubechies 4 wavelets. These results clearly indicate that the detection performance of the spline wavelets should be superior.

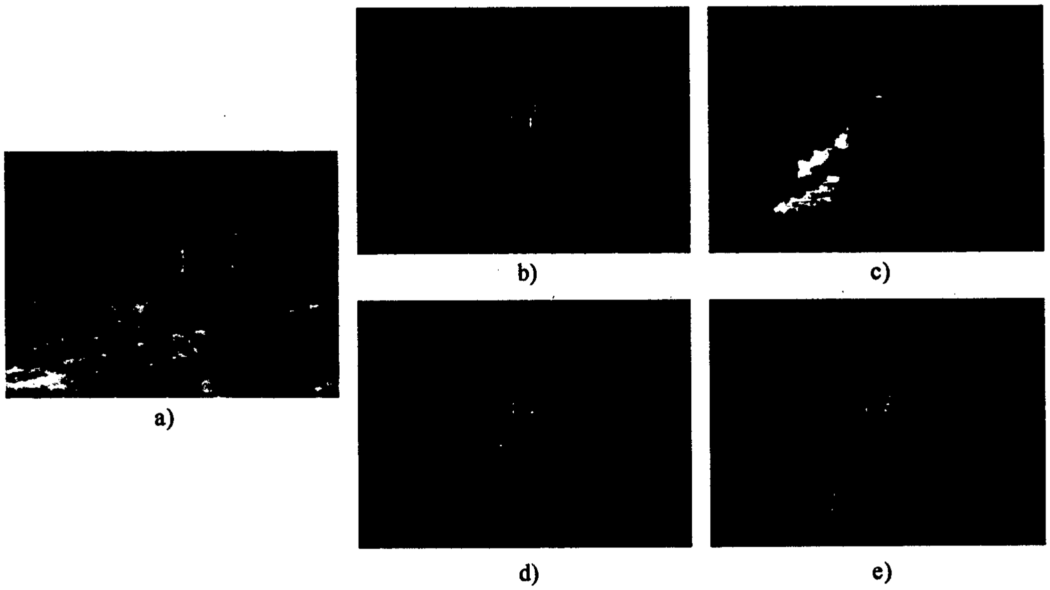

a)
b)
c)
d)
e)

Figure 39: (a) Image; (b) LH1; (c) HL1; (d) SubProd1 (e) Detection Image

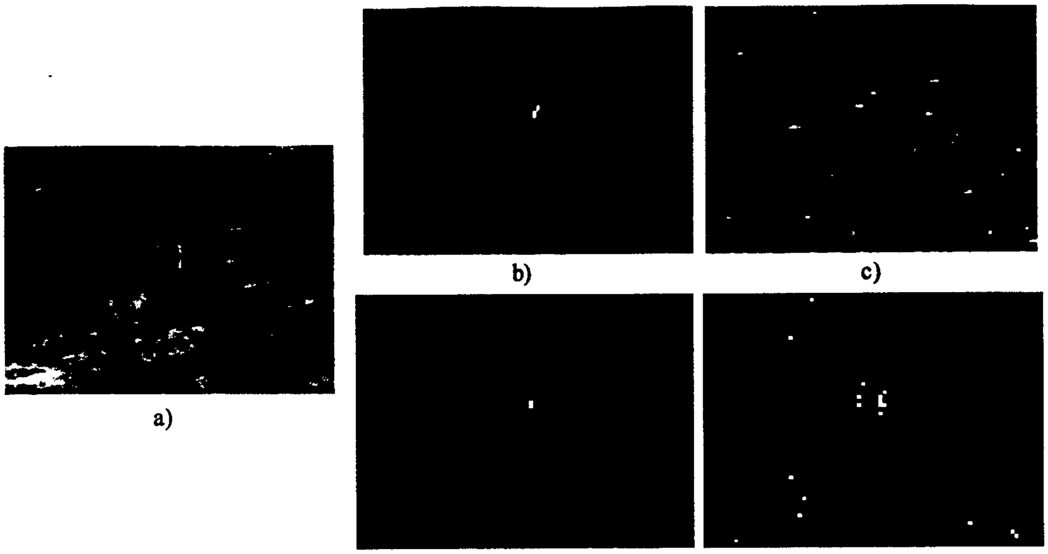


Figure 40: (a) Image; (b) LH2; (c) HL2; (d) SubProd2 (e) Detection Image

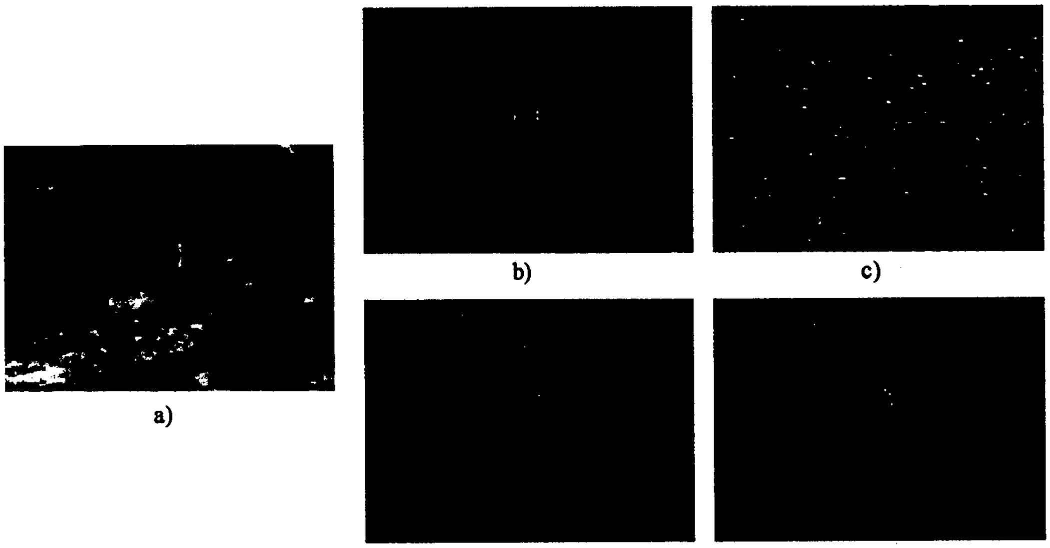


Figure 41: (a) Image; (b) LH1; (c) HL1; (d) SubProd1 (e) Detection Image

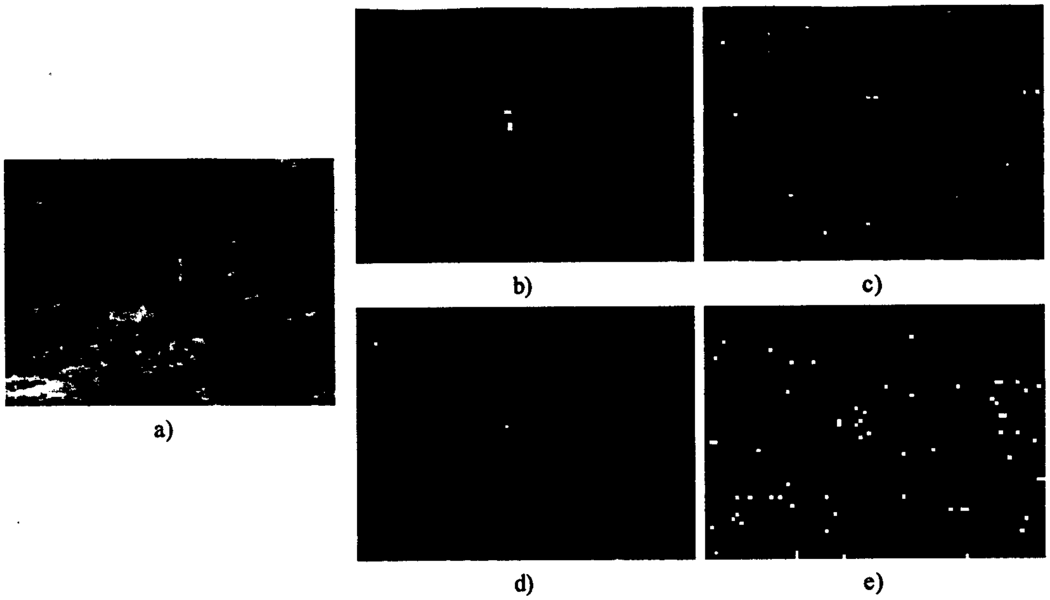


Figure 42: (a) Image; (b) LH2; (c) HL2; (d) SubProd2 (e) Detection Image

The imagery sets used for evaluation have already been described in an earlier section. We present only the best performance results for the WASP algorithm. This corresponds to two level decomposition using spline wavelets. The results are summarized in Table 4-6. While the detection results are reasonable, the false alarms are rather high. These results may be improved by fusing the output of the WASP algorithm with that of the LSVD.

	R = 600m Morning	R = 600m Noon	R = 1000m Morning	R = 1000m Noon
# Images Processed	126	126	126	126
# Targets Detected (P_D)	122 (.968)	126 (1.0)	123 (.976)	121 (.960)
# False Alarms (FA/Image)	258 (2.05)	3387 (26.88)	408 (3.24)	3846 (30.52)

	R = 1500m Morning	R = 1500m Noon	R = 2000m Morning	R = 2000m Noon
# Images Processed	126	126	126	126
# Targets Detected (P_D)	125 (.992)	116 (.921)	102 (.810)	87 (.690)
# False Alarms (FA/Image)	1302 (10.33)	7643 (60.66)	1330 (10.56)	6206 (49.25)

Table 4: Results for Yuma 1 Data for 2nd Level Spline Wavelet

	R = 600m Morning	R = 600m Noon	R = 1000m Morning	R = 1000m Noon
# Images Processed	60	60	60	60
# Targets Detected (P_D)	60 (1.0)	60 (1.0)	57 (.950)	55 (.917)
# False Alarms (FA/Image)	145 (2.42)	2243 (37.38)	114 (1.90)	3004 (50.07)

	R = 1500m Morning	R = 1500m Noon	R = 2000m Morning	R = 2000m Noon
# Images Processed	60	60	60	60
# Targets Detected (P_D)	60 (1.0)	37 (.617)	49 (.817)	24 (.400)
# False Alarms (FA/Image)	79 (1.32)	4877 (81.28)	55 (0.92)	4326 (72.10)

Table 5: Results for Yuma 2 Data for 2nd Level Spline Wavelet

	R = 600m	R = 1000m	R = 1500m
# Images Processed	16	32	31
# Targets Detected (P_D)	16 (1.0)	25 (.781)	22 (.710)
# False Alarms (FA/Image)	113 (7.06)	602 (18.81)	595 (19.19)

Table 6: Results for WSMR CFT Data for 2nd Level Spline Wavelet

2.6.2 Spline Wavelet Edge Detection

The effectiveness of the spline wavelet as an edge detector was noted during the analysis of WASP algorithm on the UCIR imagery. If an ATR system was established using the WASP algorithm, it became clear that the subbands could be processed to produce edge maps and used as an input to a model-based feature extraction and classification strategy. Further analysis was needed in order to determine the quality of edge detection given an image chip or region of interest. The final edge map produced from the first level Spline Wavelet decomposition subbands was formed as follows.

- For a given an N x N input image chip, obtain LH1, HL1 and HH1 subimages.
- Threshold LH1, HL1, and HH1 to produce binary edge maps VE1, HE1, and DE1.
- Logically OR VE1, HE1, and DE1 to produce final edge map FinalEdge1.

Sample outputs from the Yuma 2 simulation data are shown in Figure 43 for the morning time frame and Figure 44 for the noon time frame.

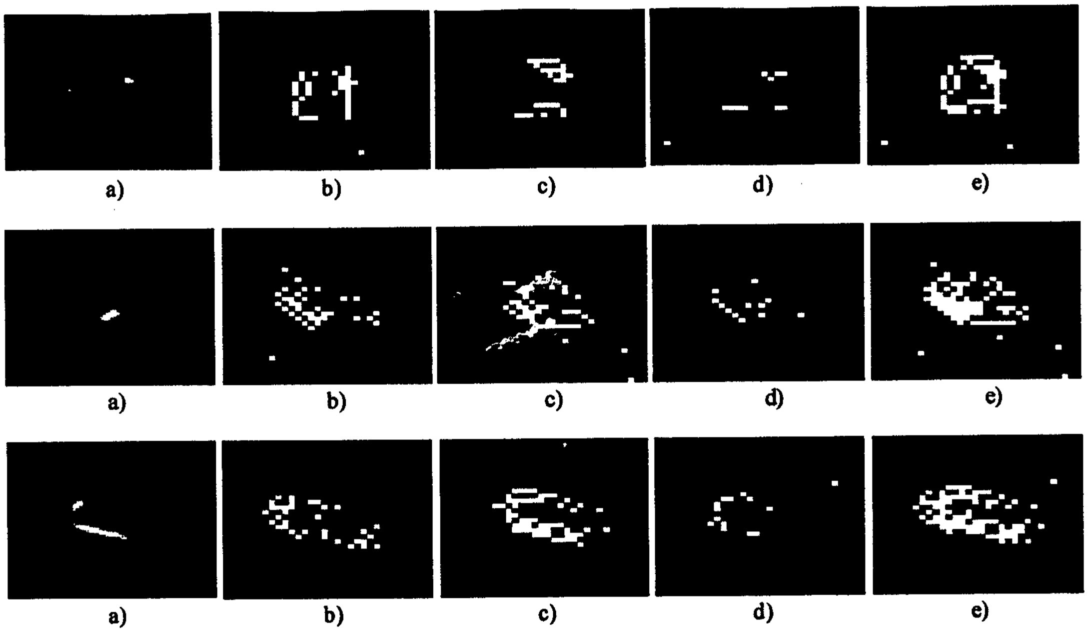


Figure 43: (a) Original Chip; (b) VE1; (c) HE1; (d) DE1 (e) FinalEdge1

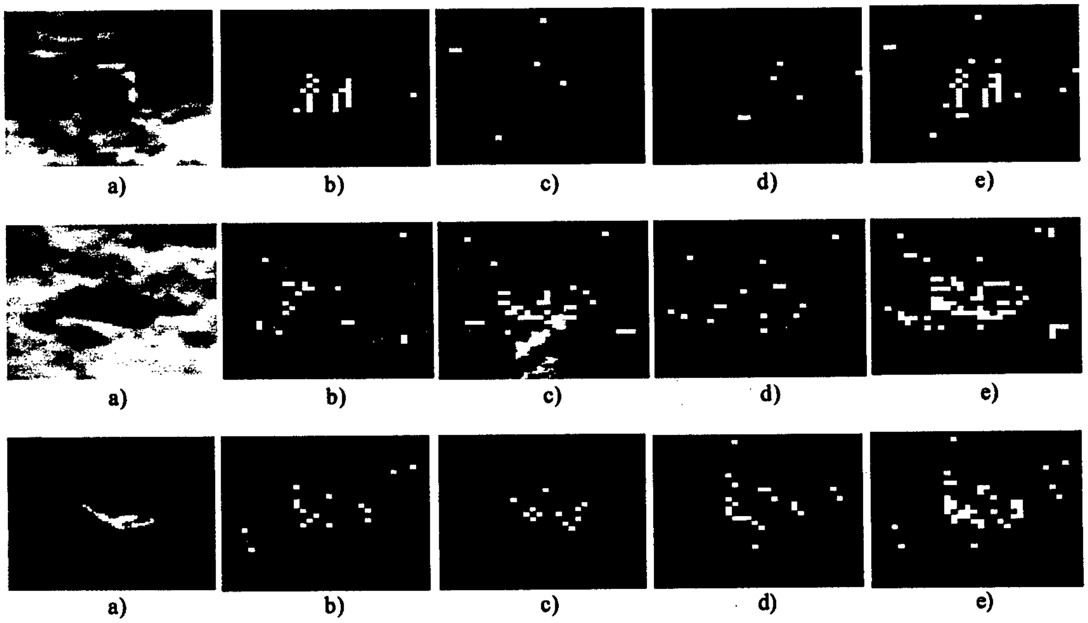


Figure 44: (a) Original Chip; (b) VE1; (c) HE1; (d) DE1 (e) FinalEdge1

<u>2.6.3 Multiscale Texture Analysis</u>

Image texture characterization has been a constant topic of research over the last 30-plus years. One of the most common statistical approaches to texture characterization is to characterize the regions with gray level properties and the spatial relationship between them through a gray level co-occurrence matrix (GLCM). Feature calculated from the GLCM can be used to discriminate between different classes of textures. The goal of this research was to identify which GLCM features could discriminate between a

T72 and BMP and if that discrimination could be enhanced at different scales of resolution. The spline wavelet was used for this study and the following six features were calculated from the GLCM generated from the original signal level, first level subbands, and second level subbands:

$$\text{Angular 2nd Moment Mean} = mean(\sum_{i=0}^{NG-1}\sum_{j=0}^{NG-1}\{p(i,j)\}^2)$$

$$\text{Sum Average Mean} = mean(\sum_{i=2}^{2NG} i p_{x+y}(i))$$

$$\text{Sum Variance Mean} = mean(\sum_{i=2}^{2NG}(i - \text{sum entropy})^2 p_{x+y}(i))$$

$$\text{Sum Variance Spread} = \max(\sum_{i=2}^{2NG}(i - \text{sum entropy})^2 p_{x+y}(i)) - \min(\sum_{i=2}^{2NG}(i - \text{sum entropy})^2 p_{x+y}(i))$$

$$\text{Sum Entropy Spread} = \max(-\sum_{i=2}^{2NG} p_{x+y}(i)\log\{p_{x+y}(i)\}) - \min(-\sum_{i=2}^{2NG} p_{x+y}(i)\log\{p_{x+y}(i)\})$$

$$\text{Info Correlatio n (2) Spread} = \max\left(1 - e^{-2(HXY_2 - HXY)}\right)^{1/2} - \min\left(1 - e^{-2(HXY_2 - HXY)}\right)^{1/2}$$

Where

$$p_{x+y}(k) = \sum_{\substack{i=0 \\ i+j=k}}^{NG-1}\sum_{j=0}^{NG-1} p(i,j),\quad k = 2,3,\ldots,2NG \qquad p_x(i) = \sum_{j=0}^{NG-1} p(i,j) \qquad p_y(j) = \sum_{i=0}^{NG-1} p(i,j)$$

$$HXY = -\sum_{i=0}^{NG-1}\sum_{j=0}^{NG-1} p(i,j)\log\{p(i,j)\} \qquad HXY_2 = -\sum_{i=0}^{NG-1}\sum_{j=0}^{NG-1} p_x(i)p_y(j)\log\{p_x(i)p_y(j)\}$$

and p(i,j) is the i^{th} row and j^{th} column entry in the GLCM.

2.7 Borrowed Strength Algorithm

The *Borrowed Strength Algorithm* (BSA) is one tool from statistical image processing that can be very useful and effective when applied to ATR. Basically, the BSA takes an input image and processes it, ultimately producing as output a segmented version of the image – one that hopefully contains the potential target(s) or ROIs (regions of interest) amongst the delineated segments. Additionally, certain segments of the image (based on their statistics or PDFs) may be highlighted as being more likely to be associated with the target.

Based on pragmatic experience in many fields (*e.g.*, medical imaging, pattern analysis, computer vision, *etc.*), some standard pre-processing steps must first be performed on the original (raw) image obtained from the sensor. Multiple options are available here, but a commonly used approach is described first; for each pixel in the raw image compute the mean μ and standard deviation σ, averaged over a distance scale "*s*". From this, compute one measure of local texture, the scalar quantity σ/μ, (the coefficient of variation) at each pixel. The resulting image is called the texture image *T*. Other pre-processing options to obtain the texture image include using tensor formulations, where combined horizontal and vertical variations have the potential of revealing coherent structure information. (These more exotic options were not examined in our study.)

The texture image *T* (input) is run through any preliminary or rudimentary segmentation algorithm to produce an initial estimate of a segmented image, called *S*. Several ways to generate S are: via watershed algorithms, other edge detectors, or even via Gaussian mixture models. Regardless of how it is obtained, the segmented image *S* will, in general, consist of *R* regions: $r = 1, \ldots R$, with each region *r* containing n_r pixels, and with known pixel coordinates (x_j, y_j) and intensities v_j for each of the regional pixels: $j=1,\ldots,n_r$.

The primary goal of the BSA is to improve upon the original segmentation of image *S* by using a statistical approach. Typically, a major problem with segmented image *S* is that it will contain an unacceptably large number of regions. Also, since many of the n_r values may be small (*e.g.*, for tiny targets), statistics of these regions are difficult to estimate accurately. The BSA addresses this problem by applying the concept that there is "strength in numbers". BSA applies statistics that are obtained from the *entire* texture image *T* to the individual regions: $r = 1, \ldots R$, obtaining "borrowed strength" estimates for the statistics (*i.e.*, probability distribution functions, PDFs) of each region. Then, using a similarity matrix built from the metric distances separating the regional PDFs, a clustering algorithm merges the regions of the initial segmented image *S* into a more useful, borrowed strength segmented image, *S'*. In this fashion the difficulties associated with poor statistics due to small sample sizes are hopefully circumvented.

The similarity matrix mentioned above is described in more detail in the literature. A listing of the routine Bld_sim_mtx is given in Appendix B. In the routine, sums of Gaussian normal functions are integrated with each other in closed form to evaluate integrated square error terms between PDFs. Appendix C contains the listing of the clustering algorithm, Cluster_alg.

There are several additional assumptions implicit in the BSA, some of them quite strong. The main one is that the underlying local densities of the different regions are mixture models of normal (Gaussian) or t-component distributions. A stronger assumption is that the underlying mixture components can be considered invariant across class, with the probability density functions differing only in their mixing coefficients. Ultimately, all of these assumptions have been tested under real case studies, checking whether the second stage segmentations *S'* using the borrowed strength methodology are indeed better than using alternative ways (such as local statistics) to improve the simplistic segmentations *S*.

2.8 Image Equalization – Local Gaussianization Algorithm

Because the contrast in the UCIR imagery is so poor, we have investigated a number of different equalization techniques. Given the extremely tight spread in pixel values, standard histogram equalization methods should fail miserably and they do. Professor Nathan Intrator from Brown University was kind enough to furnish us with an equalization algorithm that he developed for sonar signal processing. This algorithm is a local Gaussianization algorithm. We have used this algorithm for preprocessing our UCIR data sets. While the visual quality of the image can be improved (*c.f.,* Figure 38), the performance of our detection algorithms occasionally improved very marginally but was often degraded. Since the local Gaussianization algorithm contains a number of selectable parameters, we do not claim to have performance an exhaustive study. This was another time where we began our investigation near the end of the current contract and so have only been able to perform a preliminary investigation. We believe that the local Gaussianization algorithm has merit and should be pursued.

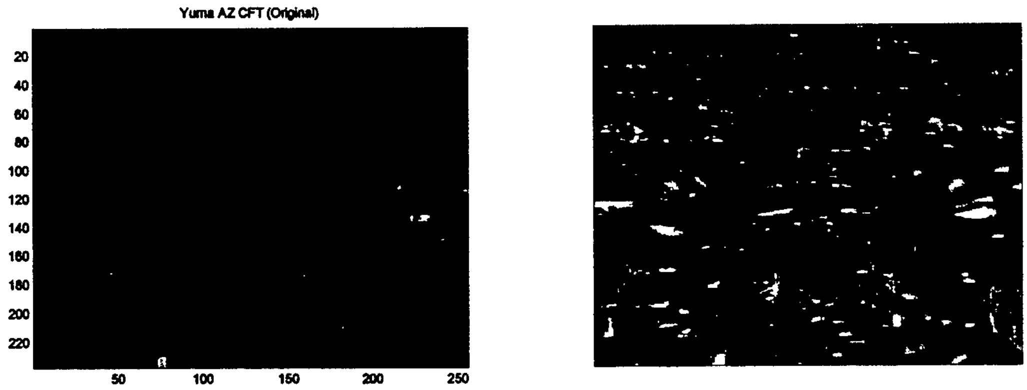


Figure 45: Preprocessed UCIR Imagery using Local Gaussianization

3.0 Detection and Classification for other Sensor Modalities

In this section, we collect some miscellaneous results corresponding to small research studies for a few additional sensor types. In particular, we address target detection for Synthetic Aperture Radar and target discrimination for cooled long wave infrared sensors for Ballistic Missile Defense Applications.

3.1 SAR Target Detection

We have also conducted a preliminary study of the LSVD algorithm for the identification of regions of interest (ROI) for Synthetic Aperture Radar (SAR). Presented below are a few Global Hawk SAR images. Some results are presented in Figure 39.

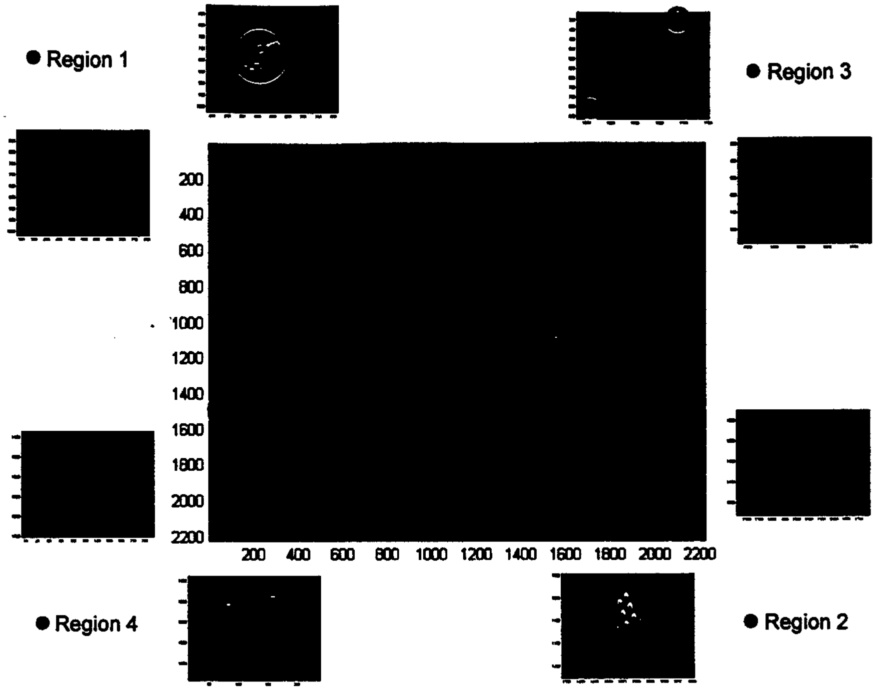


Figure 46: LSVD ROI identification for Global Hawk SAR Image

The images corresponding to the top four local singular vectors are shown in Figure 40.

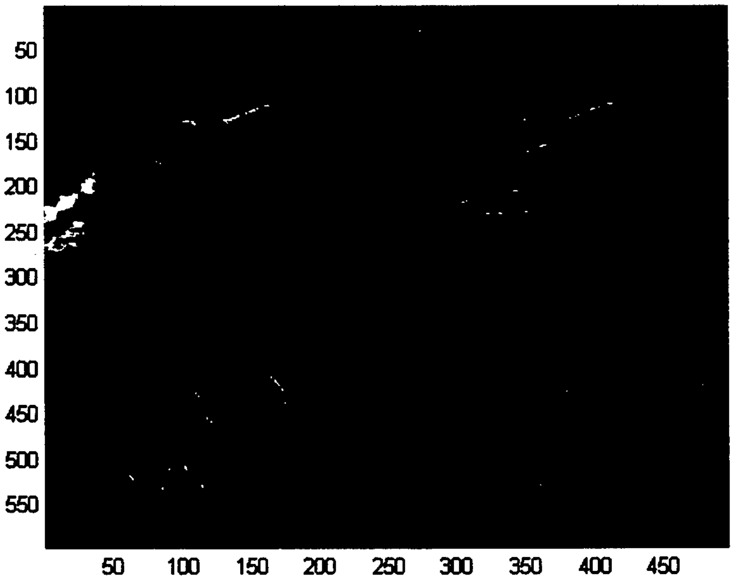


Figure 47: Eigenimages for top four singular vectors

3.2 BMD Target Discrimination

We have investigated the Local Discriminant Bases (LDB) algorithm as a feature extraction and classification algorithm for Standard Missile Lightweight Exoatmospheric Projectile (LEAP). SM-3 LEAP is the Navy's envisioned for upper tier defense. LDB is a powerful algorithmic framework that was originally developed by Coifman and Saito in 1994 as a technique for analyzing object classification problems. The LDB method selects an orthogonal basis from a large collection of orthogonal bases based on relative entropy or a similar metric. In LDB the accumulated *relative entropy* is first used as a cost function for choosing the best discriminant bases and then the individual coefficients are ranked in the order of relative entropy values.

The basis library consists of functions that are well localized in the time-frequency plane and includes discrete wavelet packets and local trigonometric bases. The localized nature of these basis functions often results in a reduced set of features that is easier to interpret and more intuitive than conventional methods. The performance of the target classifier is usually enhanced, since the LDB method reduces the dimensionality of the problem without losing important information for classification.

A large simulated database from the LEAP program was used to test different feature extraction and target classification algorithms. These simulated data sets are based on high fidelity models that have been developed by the SM-3 LEAP program. A confusion matrix for one of these data sets is shown in Figure 41. A quadratic classifier was used. While the performance illustrated here is nearly identical to that obtained using the current Fourier-based methodology, we expected to be able to demonstrate improved performance once additional threat types and countermeasures are included.

N Features	Daubechies 04		Coifman 06	
1	100.00	0.00	100.00	0.00
	39.86	60.14	39.86	60.14
2	99.40	0.60	99.20	0.80
	23.19	76.81	23.79	76.21
3	100.00	0.00	100.00	0.00
	21.62	78.38	22.34	77.66
4	99.80	0.20	100.00	0.00
	19.20	80.80	19.57	80.43
5	97.80	2.20	100.00	0.00
	18.12	81.88	19.08	80.92

(a)

	RV	Other
RV	100.00	0.00 Leakage
Other	19.08 False Alarm	80.92

(b)

Figure 48: (a) Confusion matrix for classification (b) Legend for confusion matrix

3.3 HRR Classification

In this section, we describe some very preliminary work that we are doing on target classification using high range resolution radar (HRR). At one time, HRR was considered a promising sensor modality for ATR but has pretty much been replaced as the radar sensor of choice by Synthetic Aperture Radar (SAR). In large part, this was because SAR

ATR was relatively easier and because SAR maps have a strong image-like quality that makes them attractive for applying image processing and pattern recognition techniques. Producing high-resolution SAR imagery is computationally intensive and thus extremely difficult to implement within the highly volume and power constrained world of munitions. The maturation of millimeter wave (MMW) HRR sensor technology has led to a revived interest in their use for ATR/C. An added benefit is the capability of MMW energy to penetrate rain, dust, fog and smoke. The narrow MMW beamwidth results in improved cross range resolution with the accompanying increase in the signal-to-clutter ratio (SCR). This increase in SCR is a significant characteristic due to the terrain-induced clutter present in air-to-ground scenarios. The SCR may be further improved by utilizing Doppler beam sharpening (DBS) techniques. DBS provides the capability to increase cross range resolution beyond that nominally available with standard real beam (RB) techniques.

Whether using RB or DBS, the general ATR/C approach relies on identifying features constructed from the range profiles derived. Feature estimation involves developing a suite of discriminating signature features for the targets of interest that make full use of the characteristic differences present in different classes of objects. As it is often practiced, feature identification is somewhat of an art and relies heavily on the experience of the analyst. For example, Raytheon has identified over 30 potential features. An alternative approach is to use the best bases techniques discussed in Section 3.2. Training data for two targets of interest at 360° integer aspect angles was generated for RB and DBS by processing ISAR images (*c.f.*, Figure 42).

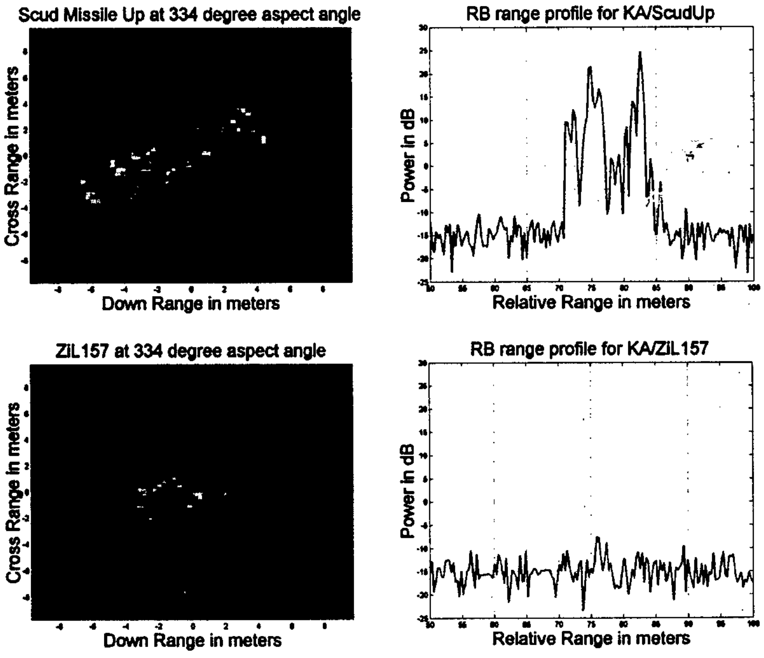


Figure 49: RB Profiles by processing ISAR images

As mentioned previously, we have identified approximately 30 relevant features for HRR sensors. Both DBS and RB data can be used as inputs to a feature estimation algorithm. Training data from all 360° aspect angles were data included. These preliminary results show feature separation between these two classes of targets. After a search though all available features, a scatter plot showing the best separation was generated and is shown in Figure 43a. The same data set was processed through LDB and a scatter plot for the top two LDB features is shown in Figure 43b. While the feature separations for the two approaches are comparable, LDB is an automated procedure. LDB also produces a related set of feature vector that can be interpreted. Since we have only just recently received the HRR data, we have not been able to perform any exhaustive studies; nonetheless, LDB appears to be a viable approach to this problem.

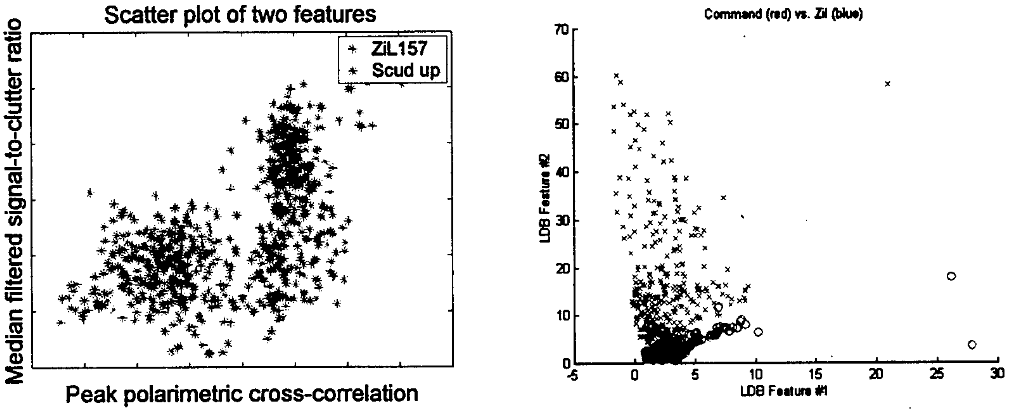


Figure 50: (a) Selected Features; (b) Top two LDB Features

We have also generated some very preliminary confusion matrices. In this analysis a simple quadratic classifier was employed and we generate confusion matrices pair-wise for our target set. The results are summarized in Table 4. Clearly, some of the performance results are not what we would desire. This is primarily due to the fact RB data is extremely aspect depend and so some preprocessing or partitioning of the data sets will be required. In addition a more powerful classifier, such as a neural network or SVM, should also improve results. Here MU is an SA12 with missile up, RDR is an SA12 radar, CMND is an SA12 command module and ZiL is a ZiL131 Truck.

Class/Truth	CMND	ZIL	Class/Truth	CMND	MU
CMND	0.9917	0.0083	CMND	0.1583	0.8417
ZIL	0.1028	0.8972	MU	0.0528	0.9472
Class/Truth	MU	ZIL	Class/Truth	CMND	RDR
MU	0.9639	0.0361	CMND	0.8417	0.1583
ZIL	0.0861	0.9139	RDR	0.6444	0.3556
Class/Truth	RDR	ZIL	Class/Truth	MU	RDR
RDR	0.2944	0.7056	MU	0.0528	0.9472
ZIL	0.0778	0.9222	RDR	0.0917	0.9083

Table 7: Pair-Wise Target Confusion Matrices

4.0 Supporting Information

4.1 Patent Applications

We had one patent application submittal during the period of performance of the contract.

1. PD-2000wTBD: **"Detection and Direction Estimation of Near Stationary Targets in Mono-static Clutter from Phase Information using Correlation and Spectrum Analysis,"** H. A. Schmitt, H.-W. Chen, G. T. David and A. A. Samuel.

4.2 Publications

We had fifteen manuscripts published over the course of our research program.

C (CONFERENCE PROCEEDINGS)
CI (CONFERENCE PROCEEDINGS, INVITED)

1. **[C] "Wavelet Based Optimization of Space-Time Adaptive Processing,"** D. C. Braunreiter, H. A. Schmitt, H.-W. Chen and G. Beylkin, ASAP '98, Boston, MA, March 12-14, 1998.

2. **[C] "Theoretical and Experimental Results on the Application of Wavelet Transforms to RF STAP and Real Time Optical Compensation,"** D. C. Braunreiter, H. A. Schmitt and H.-W. Chen, *3RD NATO-IRIS JOINT SYMPOSIUM: INNOVATION IN MILITARY AND SECURITY SENSING*, Quebec City, Quebec, October 19-23, 1998.

3. **[C] "Theoretical and Experimental Results on the Application of Wavelet Transforms to RF Space Time Adaptive Processing,"** H.A. Schmitt, L. J. Baig, H.-W. Chen and D. M. Healy, *1999 IRIS Specialty Group Meeting on Missile Defense Sensors, Environments and Algorithms*, Monterey, CA January 26-28, 1999.

4. **[C] "Adaptive Non-Uniformity Compensation Using Adaptive Feedforward Shunting and Wavelet Transforms"**, H.-W. Chen, H. A. Schmitt and D. M. Healy, 1999 IRIS Passive Sensors Meeting, , Naval Postgraduate School, Monterey, CA, February 22-24,1999.

5. **[C] "A Multi-Resolution Approach to Object Classification Using Kinematic Features,"** H.-W. Chen, H. A. Schmitt, J. G. Riddle and S. K. Mashima, *SPIE AeroSense*, Orlando, FL, April 4-9, 1999.

6. **[C] "On the Use of Space-Time Adaptive Processing and Multiresolution Data Representations for the Detection of Near-Stationary Targets in Monostatic Clutter,"** A. A. Samuel, H. A. Schmitt, G. T. David, H.-W. Chen and D. C. Braunreiter, Tri-Services Radar Conference, Monterey, CA, June 21-24, 1999.

7. **[C] "Advanced Mathematical Algorithms for Real Time Channel Equalization in Space Time Adaptive Processing Applications,"** H. A. Schmitt, M. L. Cassabaum,

H.-W. Chen, D. M. Healy and D. C. Braunreiter, Tri-Services Radar Conference, Monterey, CA, June 21-24, 1999.

8. [C] **"Applications of Local Discriminant Bases and Time-Frequency Analysis to Selected Problems in Missile Seeker Signal Processing,"** H. A. Schmitt, M. L. Cassabaum and H.-W. Chen, Hyperspectral Multispectral Simulation Workshop, Huntsville, AL, September 7-9, 1999.

9. [C] **"Detection of Near-Stationary Targets in Monostatic Clutter Using Time-Frequency Analysis and Local Discriminant Bases,"** A. A. Samuel, H. A. Schmitt, G. T. David, All Raytheon RF Symposium, November 1-4, 1999.

10. [C] **"A Fast Direct Solver Algorithm for RF Channel Equalization based on Hankel Symmetry of the Data Matrix,"** H. A. Schmitt, R. D. Rosenwald, M. A. Woolf, and H.-W. Chen, All Raytheon RF Symposium, November 1-4, 1999.

11. [C] **"Time Adaptable Continuous Wavelet Transform Analysis with Analog Signal Processing Devices,"** J. F. Scholl, D. L. Barker, H. A. Schmitt, A. A. Samuel and J. D. Langan, 2000 Processing Technology Exposition, Tucson, AZ, June 6-8, 2000.

12. [C] **"Advanced Algorithms and Architecture for Automatic Target Recognition/ Classification,"** H. A. Schmitt, 2000 Processing Technology Exposition, Tucson, AZ, June 6-8, 2000.

13. [C] **"Detection and Direction Estimation of Near-Stationary Targets in Monostatic Clutter Using RF Phase Information,"** H.-W. Chen, A. A. Samuel, H. A. Schmitt, G T. David and D. M. Healy, 46th Annual Tri-Service Radar Symposium, Colorado Springs, CO, June 28-30, 2000.

14. [C] **"Local Discriminant Bases and Time-Frequency Transforms as Discrimination Algorithms for Theater Missile Defense,"** M. L. Cassabaum, H. A. Schmitt, H.-W. Chen and J. G. Riddle, San Diego, CA, SPIE, July 2000.

15. [CI] **"On the Use of Space-Time Adaptive Processing and Time-Frequency Data Representations for the Detection of Near-Stationary Targets in Monostatic Clutter,"** D. C. Braunreiter, H.-W. Chen, M. L. Cassabaum, J. G. Riddle, A. A. Samuel, J. F. Scholl and H. A. Schmitt, 10th IEEE Workshop on Statistical Signal and Array Processing, Pocono, PA, August 14-16, 2000.

4.3 Key Program Personnel

While Raytheon had formal subcontract control over non-Raytheon team members, we conducted the program under an IPT structure. This put in place a management structure that provided a clear delineation of responsibilities, ensured communication among team members, and provided a method for the resolution of differences. Figure 44 shows the program organizational structure and illustrates a cumulative list of people that have been involved.

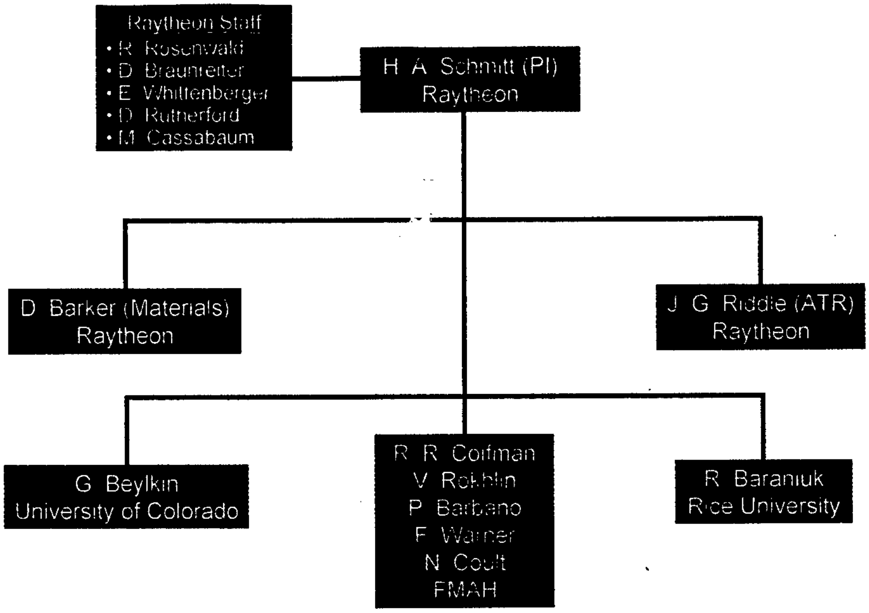


Figure 51: Advanced Mathematics IPT Structure

4.4 Program Transitions

We have developed a very successful technology transfer approach under our current DARPA contract that was based on establishing a close working relationship with Program engineers. We have had two major transition successes. The Channel Equalization algorithm described in Section 1 has been implemented in flight hardware for AMRAAM Phase 3. The fast binning algorithm for generating range-Doppler maps has been implemented in Raytheon's Tactical 6-DOF model for Air-to-Air missiles. We are currently in the process of optimizing the LSVD code as embedded software and hope to get the timing reduced sufficiently that we can get one final program transition into the NetFires PAM.

5. Acronyms

AAP	Adaptive Array Processing
ABF	Adaptive Beam Forming
ACMP	Applied and Computational Mathematics Program
ASTB	Airborne Seeker Testbed
ATA	Automatic Target Acquisition
ATR/C	Automatic Target Recognition/Classification
BSA	Borrowed Strength Algorithm
CDF	Coherence Diffusion Filtering
CFT	Captive Flight Test
CIWS	Close In Weapons Support
CPI	Coherent Processing Interval
DBS	Doppler Beam Sharpening

DOF	Degree of Freedom
DUCE	Duke University Channel Equalization
DWT	Discrete Wavelet Transform
ECM	Electronic Countermeasure
FCE	Fast Channel Equalization
FCM	Fast Covariance Matrix
FMAH	Fast Mathematical Algorithms and Hardware
FMVM	Fast Matrix Vector Multiplication
GLCM	Gray Level Co-occurrence Matrix
IRMA	Infrared Modeling Analysis
JNR	Jammer to Noise Ratio
LAM	Loitering Attack Munition
LDB	Local Discriminant Bases
LEAP	Lightweight Exoatmospheric Kill Vehicle
LSDB	Least Statistically Dependent Bases
LSVD	Local Singular Value Decomposition
MAID	Multiresolution Anisotropic Image Diffusion
MMW	Millimeter Wave
NUC	Non-uniformity Compensation
PAM	Precision Attack Munition
PC	Principle Component
PDF	Probability Distribution Function
PRI	Pulse Repetition Interval
PSD	Power Spectral Density
RB	Real Beam
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SAC	Spectral Analysis Codes
SAR	Synthetic Aperture Radar
SCNR	Signal-to-Clutter+Noise-Ratio
SCR	Signal-to-Clutter Ratio
SM	Standard Missile
SNR	Signal to Noise Ratio
SSE	Spatial Spectral Estimation
STAP	Space Time Adaptive Processing
STIG	Simulation Technology Image Generation
TERM	Tank Extended Round Munition
TSI	Terrain Scattered Interference
UAV	Unmanned Aerial Vehicle
UCIR	Uncooled Infrared
WASP	Wavelet Subband Product algorithm
WSMR	White Sands Missile Range

6. References

1. Daubeshies, I. 'Ten Lectures on Wavelets,' SIAM, Philadelphia, PA, 1992.

2. R. R. Coifman and N. Saito, 'Selection of Best Bases for Classification and Regression,' IEEE-IMS Workshop on Information and Statistics, pg. 51, 1994.
3. R. R. Coifman and N. Saito, 'Local discriminant bases and their applications,' J. Mathematical Imaging and Vision, **5**, no.4, pp.337-358, 1995, Invited paper.
4. N. Saito, 'The least statistically-dependent basis and its applications,' 32nd Asilomar Conference on Signals, Systems & Computers, **1**, pg. 732, 1998.
5. N. Saito, 'Image approximation and modeling via least statistically-dependent bases,' Pattern Recognition, vol. 34, pp.1765-1784, 2001.
6. R. R. Coifman and N. Saito, 'The local Karhunen-Loeve Bases,' Proc. IEEE International Symposium on Time-Frequency and Time-Scale Analysis, pp.129-132, IEEE Signal Processing Society.
7. D. A. Yocky, 'Image Merging and Data Fusion by Means of the Discrete Two-Dimensional Wavelet Transform,' *J. Opt. Soc. Am. A*, **12**, no. 9, pg. 1834, 1995.
8. V. Strela, *et al.*, "The application of multiwavelet filter banks to image processing," *IEEE Trans. on Image Proc.,* **8** no. 4, pg. 548, 1999.
9. C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, **20,** no. 3, pg. 273, 1995.
10. J. Weickert, *Anisotropic Diffusion in Image Processing* (1998).
11. G. H. Cottet and M. El Ayyadi, "Nonlinear PDE operators with memory terms for image processing", *Proceedings of the IEEE International Conference on Image Processing* (ICIP-96), Lausanne, Switzerland, 16-19 September 1996, Vol. 1 (1996) 481-483.
12. R. Tumbar and D. J. Brady, 'Sensor plane processing for multiplex imaging,' 2000 Southwest Symposium on Mixed-Signal Design, Pp. 1-6, 2000.
13. C. E. Priebe, D. J., Marchette and G. W. Rogers, 'Segmentation of random fields via borrowed strength density estimation,' IEEE Trans. on PAMI, **19,** no. 5, pp. 494-499, 1997.
14. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.
15. http://www-stat.stanford.edu/~wavelab/
16. H. I. Hayes, C. E. Priebe, G. W. Rogers, D. L. Marchette, J. L. Solka and R. A. Lorey, 'Improved texture discrimination and image segmentation with boundary incorporation', **2485,** no. 7, pp. 260-267,1995.
17. Y. Wan and R. D. Nowak A Wavelet-Based Statistical Model for Image Restoration in Proc. IEEE International Conference on Image Processing - ICIP'01, Thessaloniki, Greece, Oct. 2001.
18. V. Venkatachalam, H. Choi, R.G. Baraniuk Multiscale SAR Image Segmentation using Wavelet-domain Hidden Markov Tree Models Proceedings of the SPIE 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls, Algorithms for Synthetic Aperture Radar Imagery VII, 24-28 April 2000, Orlando, FL.
19. H. Choi and R. G. Baraniuk, Multiscale texture segmentation using wavelet-domain hidden Markov models, Proc. 32nd Asilomar Conference, November 1998.
20. R. G. Baraniuk, D. L. Jones, A Signal-Dependent Time-Frequency Representation: Optimal Kernel Design, IEEE Transactions on Signal Processing, **41**, no. 4, pp. 1589-1602, April 1993.

21. A. H. Tewfik and M. Kim, 'Fast Positive Definite Linear System Solvers,' IEEE Trans. on Signal Processing, V. 42, no. 3, pp. 572-585, 1995
22. S. Hosur and A. Tewfik 'Wavelet Transform Domain Adaptive FIR Filtering,' IEEE transactions on signal processing, **45**, no. 3, pp. 617, 1997.
23. S. Kadambe and Y. Owechko, 'Computation Reduction in Space Time Adaptive Processing (STAP) of Radar Signals Using Orthogonal Wavelet Decompositions,' Asilomar Conference on Signals Systems and Computers, **1**, pp. 641-645, 2000.
24. R. M. Haralick, 'Propagating Covariance in Computer Vision,' International journal of pattern recognition and artificial intelligence, **10**, no. 5, pp. 561, 1996.
25. T. J. Burns, S. K. Rogers, M. E. Oxley, D. W. Ruck, 'A Wavelets Multiresolution Analysis for Spatio-Temporal Signals,' IEEE Transactions on Aerospace and Electronics Systems, V. 32, no. 2, pp 628-649, 1996
26. S. Haykin and D. J. Thomson, 'Signal Detection in a Nonstationary Environment Reformulated as an Adaptive Pattern Classification Problem,' Proceedings of the ieee, **86**, no. 11, pp. 2325, 1998.
27. D. P. Casasent and D. M. Weber, 'Quadratic Gabor Filters for Object Detection,' IEEE Transactions on Image Processing, **10**, no. 2, pp. 218-230, 2001.
28. R. Klemm, 'Introduction to space-time adaptive processing,' Electronics & communications engineering journal, **11**, no. 1, pp. 5, 1999.
29. G. Beylkin, N. Coult and M. J. Mohlenkamp, 'Fast Spectral Projection Algorithms for Density-Matrix Computations,' Journal of computational physics, **152**, no. 1, pp. 32 1999.
30. G. Beylkin, R. Coifman and V. Rokhlin, "Fast Wavelet Transforms and Numerical Algorithms I," Commun. Pure Applicat. Math., V. 44, pp. 141-183, 1991.
31. R. Haimi-Cohen and A. Cohen, 'Gradient Type Algorithms for Partial Singular Value Decomposition,' IEEE Trans. On PAMI, **9**, no.1, pp. 137-142, 1987
32. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, New Jersey, 2002.
33. D. A. Yocky, 'Image Merging and Data Fusion by Means of the Discrete Two-Dimensional Wavelet Transform,' *J. Opt. Soc. Am. A*, **12**, no. 9, pg. 1834, 1995.
34. V. Strela, *et al.*, 'The application of multiwavelet filter banks to image processing,' *IEEE Trans. on Image Proc.*, **8** no. 4, pg. 548, 1999.
35. S. C. Zhu and D. Mumford, 'GRADE: Gibbs Reaction and Diffusion Equaition,' IEEE Trans. PAMI, **19**, no. 11, 1997.
36. C. Cortes and V. Vapnik, 'Support-Vector Networks,' Machine Learning, **20,** no. 3, pg. 273, 1995.
37. D. L. Gines, G. Beylkin and J. Dunn, 'LU Factorization of Non-Standard Forms and Direct Multiresolution Solvers,' Applied and Computational Harmonic Analysis, 5, pp. 156-201, 1998.
38. G. Beylkin, 'On factored FIR approximation of IIR filters, Applied and Computational Harmonic Analysis, 2, pp. 293-298, 1995.
39. G. Beylkin, 'Fast and accurate computation of the Fourier transform of an image,' Proceedings of SPIE, **2277**, pp. 244-252, 1994.
40. A. Rihaczek and S Hershkowitz, *Radar Resolution and Complex-Image Analysis*, Artech House, 1996.
41. Ulaby and Dobson, *Handbook of Radar Scattering Statistics for Terrain*, 1985.

42. M.V. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software*, A K Peters, Ltd., 1994.
43. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.
44. D. L. Donoho, 'Denoising via soft thresholding,' IEEE Transactions on Information Theory, v. 41, no. 3, pp 613-627, 1995.
45. G. Beylkin, R. Coifman and V. Rokhlin, "Fast Wavelet Transforms and Numerical Algorithms I," Commun. Pure Applicat. Math., V. 44, pp. 141-183, 1991.
46. S. Chawla, *Novel Compression Techniques with Applications in Medial Image Acquisition and Image Storage*, Ph.D. Dissertation, Dartmouth College, 1998.
47. S. Haykin and T. Bhattacharya, 'Wigner-Ville distribution: an important functional block for radar target detection in clutter,' *Signals, Systems and Computers*, A. Singh Editor, Twenty-Eighth Asilomar Conference, pp. 68–72, vol.1, 31 Oct.-2 Nov. 1994.
48. N. C. Currie and C. E. Brown, *eds., Principles and Applications of Millimeter-Wave Radar,* Artech House, Norwood, MA, 1987.
49. L. M. Novak and S. R. Hesse, 'Optimal Polarizations for Radar Detection and Recognition of Targets in Clutter,' Radar Conference, Record of the 1993 IEEE National, 1993 pp. 79-83, 1993.
50. D. K. Barton, *Modern Radar System Analysis*, Artech House, Norwood, MA, 1988.
51. I. S. Gradshteyn and I. M. Ryzhik, *Tables of Integrals, Series and Products*, Academic Press, 1994.
52. U. Ndili, R. Nowak, R. Baraniuk, H. Choi and M. Figueiredo *"Coding Theoretic Approach to SAR Image Segmentation"*, to appear in the Proceedings of the SPIE 15th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls, Algorithms for Synthetic Aperture Radar Imagery VII, 16-20 April 2001, Orlando, Florida.
53. U. Ndili, R. Nowak and M. Figueiredo *"Coding Theoretic Approach to Image Segmentation"*, ICIP, October 2001.
54. T. Joachims, *"Large Scale Support Vector Machines"* to appear in Advances in Kernel Methods – Support vector Learning. http://www-ai.cs.uni-dormund.de/svm_light.

Appendix A: DUCE Algorithm C-Code

```
/***************************************************************
DISPLACEMENT RANK DECOMPOSITION ALGORITHM

Verify Xiaobai's Cholesky Decomposition Algorithm using real data.

Requires data_newPNcode.c, FormCovFFT.c, DispRank.c, DispL02.c
fft.c, new_approach.h, and the data files specified in data_newPNcode.c

Matlab code: James Thombrue, June 18, 2001
C code: David Zaugg, Nov 1, 2001
***************************************************************/

#include "new_approach.h"

int main()
{
        struct COMPLEX *EQ_data;
        struct COMPLEX *ref, *y;
        long cols, rows;
        int P, M;
```

```
int n, i, channel;
struct COMPLEX yhead[(N-1)];
struct COMPLEX ytail[(N-1)];
struct COMPLEX C[N];
struct COMPLEX B[N];
struct COMPLEX F[N][4];
int Sigma[4];
struct COMPLEX L[N][N];
FILE *handle_Lfile;

//load the channel data into EQ_data (10 channels, length 999)
//data_newPNcode
EQ_data = data_newPNcode("PNcodefilelist.txt", EQ_data, &cols, &rows);

//ref = EQ_data(:,1);
ref = (struct COMPLEX*)malloc(8*2*cols);
for(n = 0; n < cols; n += 1)
{
        (ref[n]).real = (EQ_data[n]).real;
        (ref[n]).imag = (EQ_data[n]).imag;
}

//P = length(ref);
P = cols;
//N = 16;
//M = P - N + 1;
M = P - N + 1;

//channel = 2;
channel = 2;

//y = EQ_data(:,channel);
y = (struct COMPLEX*)malloc(8*2*(cols+1));
for(n = 0; n < cols; n += 1)
{
        (y[n]).real = (EQ_data[n + cols*(channel - 1)]).real;
        (y[n]).imag = (EQ_data[n + cols*(channel - 1)]).imag;
}

//yhead = y(1:N-1);
for(n = 0; n < (N-1); n += 1)
{
        (yhead[n]).real = (y[n]).real;
        (yhead[n]).imag = (y[n]).imag;
}

//ytail = y(M+1:P);
for(n = 0; n < (P-M); n += 1)
{
        (ytail[n]).real = (y[n+M]).real;
        (ytail[n]).imag = (y[n+M]).imag;
}

//[C, B] = FormCovFFT(N, y, ref);
FormCovFFT(cols, y, ref, C, B);

//[F, Sigma] = DispRank(C, yhead, ytail);
DispRank(C, yhead, ytail, F, Sigma);

//L = DispL02(F, Sigma);
DispL02(F, Sigma, L);

//This section writes the matrix L to Lfile.txt
handle_Lfile = fopen("Lfile.txt", "w");
fprintf(handle_Lfile, "Channel %d\n\n", channel);
for(n = 0; n < N; n += 2)
{
        fprintf(handle_Lfile, "\n\nColumns %d and %d\n\n", n+1, n+2);
```

```
			for(i = 0; i < N; i += 1)
			{
					fprintf(handle_Lfile, "%.14f %.14fi\t\t%.14f %.14fi\n", (L[i][n]).real, (L[i][n]).imag,
(L[i][n+1]).real, (L[i][n+1]).imag);
			}
		}
		fclose(handle_Lfile);


		free(EQ_data);
		free(ref);
		free(y);
		return 0;
}
/**********************************************************
FormCovFFT.c
Forms first column of covariance matrix using FFT
David Zaugg
Nov 1, 2001

Inputs:
N						size of covariance matrix
len_y				length of y
y						one of the columns of the data matrix depending on channel
						column vector
ref						first column of the data matrix
						rhs of LS system

Outputs:
a						first column of covariance matrix A
b						not used, rhs of the covariance system

Xiaobai Sun, April 20, 2001
**********************************************************/

#include "new_approach.h"

void FormCovFFT(int len_y, struct COMPLEX* y, struct COMPLEX* c, struct COMPLEX a[], struct COMPLEX b[])
{
		int m, i, j, k, l, q;
		struct COMPLEX p2[N], h1[N], h2[N],  temp;
		double tempa[4*N], tempb[4*N], temph[4*N];
		int n = N;

		//m = length(y) - n + 1;
		m = len_y - n + 1;
		//y(m+n) = 0;
		(y[m+n-1]).real = 0;
		(y[m+n-1]).imag = 0;

		//if(n < 1 | m < 2*n)
		//		error('FormCovFFT: n or m is too small:');
		//end
		if(n < 1 || m < 2*n)
		{
				printf("Error in FormCovFFT\n");
				printf("FormCovFFT: n or m is too small:\n");
				exit(1);
		}

		//a = zeros(n,1);
		//b = zeros(n,1);
		//p2 = zeros(n,1);
		for(i = 0; i < n; i += 1)
		{
				(a[i]).real = 0;
				(a[i]).imag = 0;
				(b[i]).real = 0;
```

```
        (b[i]).imag = 0;
        (p2[i]).real = 0;
        (p2[i]).imag = 0;
}

//k = floor(m/n);
k = (int)floor((double)(m/n));  //number of segments of length n
//l = m - n*k;
l = m - n*k;                                                   //the remainder

// use FFT on k full segments

//index = 1:n;
//h1 = y(index);
for(i = 0; i < n; i += 1)
{
        (h1[i]).real = (y[i]).real;
        (h1[i]).imag = (y[i]).imag;
}

//for i = 1:k
for(i = 0; i < k; i += 1)
{
        //h2 = y(i*n + index);
        q = 0;
        for(j = (i + 1)*n; j <= ((i + 1)*n + n - 1); j += 1)
        {
                (h2[q]).real = (y[j]).real;
                (h2[q]).imag = (y[j]).imag;
                q += 1;
        }
        //[h1; h2]*n*2
        q = 0;
        for(j = 0; j < 4*n; j += 2)
        {
                if(j < 2*n)
                {
                        temph[j] = (h1[q]).real*n*2;
                        temph[j+1] = (h1[q]).imag*n*2;
                }
                else
                {
                        temph[j] = (h2[q-n]).real*n*2;
                        temph[j+1] = (h2[q-n]).imag*n*2;
                }
                q += 1;
        }
        //temph = fft([h1; h2]*n*2);
        four1(temph - 1, 2*n, 1); //fft when 3rd arg = 1
        //[conj(h1); p2]
        q = 0;
        for(j = 0; j < 4*n; j += 2)
        {
                if(j < 2*n)
                {
                        tempa[j] = (h1[q]).real;
                        tempa[j+1] = -(h1[q]).imag;
                }
                else
                {
                        tempa[j] = (p2[q-n]).real;
                        tempa[j+1] = (p2[q-n]).imag;
                }
                q += 1;
        }
        //ifft([conj(h1); p2])
        four1(tempa - 1, 2*n, -1); //ifft with 3rd arg = -1
        //must multiply by 1/N for ifft
```

```
for(j = 0; j < 4*n; j += 1)
        tempa[j] = tempa[j]*(double)1.0/((double)2.0*(double)n);
//ifft([conj(h1); p2]).*temph
for(j = 0; j < 4*n; j += 2)
{
        temp.real = tempa[j]*temph[j] - tempa[j+1]*temph[j+1];
        temp.imag = tempa[j]*temph[j+1] + tempa[j+1]*temph[j];
        tempa[j] = temp.real;
        tempa[j+1] = temp.imag;
}
//tempa = ifft(ifft([conj(h1); p2]).*temph);
four1(tempa - 1, 2*n, -1);
//must multiply by 1/N for ifft
for(j = 0; j < 4*n; j += 1)
        tempa[j] = tempa[j]*(double)1.0/((double)2.0*(double)n);
//a = a + tempa(index);
q = 0;
for(j = 0; j < 2*n; j += 2)
{
        (a[q]).real = (a[q]).real + tempa[j];
        (a[q]).imag = (a[q]).imag + tempa[j+1];
        q += 1;
}
//conj(c((i-1)*n+index))
q = 0;
for(j = i*n; j <= i*n + n - 1; j += 1)
{
        tempb[q] = (c[j]).real;
        tempb[q+1] = -(c[j]).imag;
        q += 2;
}
//[conj(c((i-1)*n+index));p2]
q = 0;
for(j = 2*n; j < 4*n; j += 2)
{
        tempb[j] = (p2[q]).real;
        tempb[j+1] = (p2[q]).imag;
        q += 1;
}
//ifft([conj(c((i-1)*n+index));p2]);
four1(tempb - 1, 2*n, -1);
//must multiply by 1/N for ifft
for(j = 0; j < 4*n; j += 1)
        tempb[j] = tempb[j]*(double)1.0/((double)2.0*(double)n);
//ifft([conj(c((i-1)*n+index));p2]).*temph;
for(j = 0; j < 4*n; j += 2)  //.*
{
        temp.real = tempb[j]*temph[j] - tempb[j+1]*temph[j+1];
        temp.imag = tempb[j]*temph[j+1] + tempb[j+1]*temph[j];
        tempb[j] = temp.real;
        tempb[j+1] = temp.imag;
}
//ifft(ifft([conj(c((i-1)*n+index));p2]).*temph);
four1(tempb - 1, 2*n, -1);
//must multiply by 1/N for ifft
for(j = 0; j < 4*n; j += 1)
        tempb[j] = tempb[j]*(double)1.0/((double)2.0*(double)n);
//b = b + tempb(index);
q = 0;
for(j = 0; j < 2*n; j += 2)
{
        (b[q]).real = (b[q]).real + tempb[j];
        (b[q]).imag = (b[q]).imag + tempb[j+1];
        q += 1;
}
//h1 = h2;
for(j = 0; j < n; j += 1)
{
```

```
                                (h1[j]).real = (h2[j]).real;
                                (h1[j]).imag = (h2[j]).imag;
                        }
        }

        //if(l ~= 0)
        if(l != 0)
        {
                //direct computation of the partial segment
                //index = (n*k+1):m;
                //for i = 1:n
                for(i = 0; i < n; i += 1)
                {
                        //y(index)'*y(i-1+index);
                        temp.real = 0;
                        temp.imag = 0;
                        for(j = n*k; j < m; j += 1)
                        {
                                temp.real = temp.real + (y[j]).real*(y[j+i]).real + (y[j]).imag*(y[j+i]).imag;
                                temp.imag = temp.imag + (y[j]).real*(y[j+i]).imag - (y[j]).imag*(y[j+i]).real;
                        }
                        //a(i) = a(i) + y(index)'*y(i-1+index);
                        (a[i]).real = (a[i]).real + temp.real;
                        (a[i]).imag = (a[i]).imag + temp.imag;
                        //y(index)'*c(i-1+index);
                        temp.real = 0;
                        temp.imag = 0;
                        for(j = n*k; j < m; j += 1)
                        {
                                temp.real = temp.real + (y[j]).real*(c[j+i]).real + (y[j]).imag*(c[j+i]).imag;
                                temp.imag = temp.imag + (y[j]).real*(c[j+i]).imag - (y[j]).imag*(c[j+i]).real;
                        }
                        //b(i) = b(i) + y(index)'*c(i-1+index);
                        (b[i]).real = (b[i]).real + temp.real;
                        (b[i]).imag = (b[i]).imag + temp.imag;
                }
        }

        //a = conj(a);
        for(i = 0; i < n; i += 1)
                (a[i]).imag = -(a[i]).imag;

        //a(1) = real(a(1));
        (a[0]).imag = 0;        //diagonal element, numerical reason

        //b = conj(b);
        for(i = 0; i < n; i += 1)
                (b[i]).imag = -(b[i]).imag;

        return;
}
/***************************************************************
DispRank.c
Form the displacement structure
David Zaugg
Nov 1, 2001

Inputs:
fcol                                    first column/row of covariance matrix
yhead and ytail                 from the parts of the data matrix that do not contribute
                                                to every member of the covariance matrix
                                                the first and last n-1 elements of the data vector

outputs:
F                                               displacement columns
                                                used by DispL02 to form the Cholesky of the covariance matrix

Sigma                                   displacement structure
                                                used by DispL02 to form the Cholesky of the covariance matrix
```

```
A = ZAZ' + F/Sigma F';

Xiaobai Sun 04/20/01
*************************************************************/

#include "new_approach.h"

void DispRank(struct COMPLEX fcol[], struct COMPLEX yhead[], struct COMPLEX ytail[], struct COMPLEX F[][Q], int Sigma[])
{
        int i, j;
        struct COMPLEX alpha;

        //F = zeros(n,4);
        for(i = 0; i < N; i += 1)
        {
                for(j = 0; j < 4; j += 1)
                {
                        (F[i][j]).real = 0.0;
                        (F[i][j]).imag = 0.0;
                }
        }

        //alpha = sqrt(fcol(1));
        alpha.real = sqrt(sqrt((fcol[0]).real*(fcol[0]).real +
(fcol[0]).imag*(fcol[0]).imag))*cos(0.5*atan2((fcol[0]).imag,(fcol[0]).real));
        alpha.imag = sqrt(sqrt((fcol[0]).real*(fcol[0]).real +
(fcol[0]).imag*(fcol[0]).imag))*sin(0.5*atan2((fcol[0]).imag,(fcol[0]).real));

        //F(:,1) = fcol/alpha;
        //F(:,2) = F(:,1);
        for(i = 0; i < N; i += 1)
        {
                (F[i][0]).real = ((fcol[i]).real*alpha.real + (fcol[i]).imag*alpha.imag)/(alpha.real*alpha.real +
alpha.imag*alpha.imag);
                (F[i][0]).imag = ((fcol[i]).imag*alpha.real - (fcol[i]).real*alpha.imag)/(alpha.real*alpha.real +
alpha.imag*alpha.imag);
                (F[i][1]).real = (F[i][0]).real;
                (F[i][1]).imag = (F[i][0]).imag;
        }
        //F(1,2) = 0;
        (F[0][1]).real = 0.0;
        (F[0][1]).imag = 0.0;


        //F(2:n,3) = conj(ytail);
        //F(2:n,4) = conj(yhead);
        for(i = 1; i < N; i += 1)
        {
                (F[i][2]).real = (ytail[i-1]).real;
                (F[i][2]).imag = -(ytail[i-1]).imag;
                (F[i][3]).real = (yhead[i-1]).real;
                (F[i][3]).imag = -(yhead[i-1]).imag;
        }

        //Sigma = [ 1, -1, 1, -1 ];
        Sigma[0] = 1;
        Sigma[1] = -1;
        Sigma[2] = 1;
        Sigma[3] = -1;

        return;
}
/*****************************************************************
DispL02.c
Derive the Cholesky L from the displacement structure ( F, Sigma )
David Zaugg
Nov 1, 2001
```

```
Inputs:
F                                               displacement columns
                                                used by DispL02 to form the Cholesky of the covariance matrix
Sigma                                   displacement structure
                                                used by DispL02 to form the Cholesky of the covariance matrix

Outputs:
L                                               Cholesky decomposition

Arithmetic complexity :
  6n(n-1)   complex multiplications
  3n(n-1)   complex additions
  3(n-1)    square roots
In comparison to the dense Cholesky decomposition
  (n^3-n)/6   complex multiplications
  (n^3-n)/6   complex additions
  n         square roots

Xiaobai Sun % 04/20/01
*******************************************************************/

#include "new_approach.h"

void DispL02(struct COMPLEX F[][Q], int Sigma[], struct COMPLEX L[][N])
{
        struct COMPLEX temp[N][4];
        int index[4] = {-1, -1, -1, -1};
        int i, j, k, q, p;
        struct COMPLEX c1, s1, c2, s2, c, s, r;
        double r1, r2;
        struct COMPLEX matrix[2][4];
/*
        for(i = 0; i < N; i += 1)
        {
                for(j = 0; j < 4; j += 1)
                {
                        (F[i][j]).real = (H[i][j]).real;
                        (F[i][j]).imag = (H[i][j]).imag;
                }
        }
*/

        for(i = 0; i < N; i += 1)
        {
                for(j = 0; j < 4; j += 1)
                {
                        (temp[i][j]).real = 0;
                        (temp[i][j]).imag = 0;
                }
        }

        //[n, q] = size(F);
        //N, Q defined in new_approach.h

        //if(q > 4)
        //      error('DispL : displacement rank > 4 ?');
        //end
        if(Q > 4)
        {
                printf("Error in DispL02\n");
                printf("DispL : displacement rank > 4 ?\n");
                exit(1);
        }

        //index = find(Sigma == 1);
        q = 0;
        for(i = 0; i < 4; i += 1)
        {
```

```
		if(Sigma[i] == 1)
		{
			index[q] = i;
			q += 1;
		}
}

//p = length(index);
p = 0;
for(i = 0; i < 4; i += 1)
{
		if(index[i] != -1)
			p += 1;
}

//if(p == q)
//		error('DispL : displacement signature is definite ?');
//end
if(p == Q)
{
		printf("Error in DispL02\n");
		printf("DispL : displacement signature is definite ?\n");
		exit(1);
}

//L = zeros(n);
for(i = 0; i < N; i += 1)
{
		for(j = 0; j < N; j += 1)
		{
			(L[i][j]).real = 0.0;
			(L[i][j]).imag = 0.0;
		}
}

//index = [index find(Sigma == -1)];
q = 0;
while(index[q] != -1)
		q += 1;
for(i = 0; i < 4; i += 1)
{
		if(Sigma[i] == -1)
		{
			index[q] = i;
			q += 1;
		}
}

//F = F(:,index);
for(i = 0; i < Q; i += 1)  //permutation
{
		for(j = 0; j < N; j += 1)
		{
			(temp[j][i]).real = (F[j][index[i]]).real;
			(temp[j][i]).imag = (F[j][index[i]]).imag;
		}
}
for(i = 0; i < q; i += 1)  //the first column of L
{
		for(j = 0; j < N; j += 1)
		{
			(F[j][i]).real = (temp[j][i]).real;
			(F[j][i]).imag = (temp[j][i]).imag;
		}
}

//L(:,1) = F(:,1);
for(i = 0; i < N; i += 1)
```

```
{
        (L[i][0]).real = (F[i][0]).real;
        (L[i][0]).imag = (F[i][0]).imag;
}

//for j = 2:n
for(j = 1; j < N; j += 1)
{
        //index = j:n;
        //F(index, 1) = F(index-1,1);
        for(i = N - 1; i >= j; i -= 1)  //Shift down in the first column
        {
                (F[i][0]).real = (F[i-1][0]).real;
                (F[i][0]).imag = (F[i-1][0]).imag;
        }

        // ... positive definite transform

        //c1 = F(j,1);
        c1.real = (F[j][0]).real;
        c1.imag = (F[j][0]).imag;
        //s1 = F(j,2);
        s1.real = (F[j][1]).real;
        s1.imag = (F[j][1]).imag;
        //r1 = sqrt(abs(c1)^2 + abs(s1)^2);
        r1 = sqrt(c1.real*c1.real + c1.imag*c1.imag + s1.real*s1.real + s1.imag*s1.imag);

        //c1 = c1/r1;
        //s1 = s1/r1;
        c1.real = c1.real/r1;
        c1.imag = c1.imag/r1;
        s1.real = s1.real/r1;
        s1.imag = s1.imag/r1;

        //F(index,1:2) = F(index,1:2)*[c1', -s1; s1', c1];
        (matrix[0][0]).real = c1.real;
        (matrix[0][0]).imag = -c1.imag;
        (matrix[0][1]).real = -s1.real;
        (matrix[0][1]).imag = -s1.imag;
        (matrix[1][0]).real = s1.real;
        (matrix[1][0]).imag = -s1.imag;
        (matrix[1][1]).real = c1.real;
        (matrix[1][1]).imag = c1.imag;
        for(i = j; i < N; i += 1)
        {
                for(k = 0; k < 2; k += 1)
                {
                        (temp[i][k]).real = (F[i][0]).real*(matrix[0][k]).real - (F[i][0]).imag*(matrix[0][k]).imag
+ (F[i][1]).real*(matrix[1][k]).real - (F[i][1]).imag*(matrix[1][k]).imag;
                        (temp[i][k]).imag = (F[i][0]).imag*(matrix[0][k]).real +
(F[i][0]).real*(matrix[0][k]).imag + (F[i][1]).imag*(matrix[1][k]).real + (F[i][1]).real*(matrix[1][k]).imag;
                }
        }
        for(i = j; i < N; i += 1)
        {
                for(k = 0; k < 2; k += 1)
                {
                        (F[i][k]).real = (temp[i][k]).real;
                        (F[i][k]).imag = (temp[i][k]).imag;
                }
        }

        // ... negative definite transform

        //c2 = F(j,3);
        c2.real = (F[j][2]).real;
        c2.imag = (F[j][2]).imag;
        //s2 = F(j,4);
```

```
        s2.real = (F[j][3]).real;
        s2.imag = (F[j][3]).imag;
        //r2 = sqrt(abs(c2^2 + abs(s2)^2));
        r2 = sqrt(c2.real*c2.real + c2.imag*c2.imag + s2.real*s2.real + s2.imag*s2.imag);

        //c2 = c2/r2;
        //s2 = s2/r2;
        c2.real = c2.real/r2;
        c2.imag = c2.imag/r2;
        s2.real = s2.real/r2;
        s2.imag = s2.imag/r2;

        //F(index,3:4) = F(index,3:4)*[c2', -s2; s2', c2];
        (matrix[0][2]).real = c2.real;
        (matrix[0][2]).imag = -c2.imag;
        (matrix[0][3]).real = -s2.real;
        (matrix[0][3]).imag = -s2.imag;
        (matrix[1][2]).real = s2.real;
        (matrix[1][2]).imag = -s2.imag;
        (matrix[1][3]).real = c2.real;
        (matrix[1][3]).imag = c2.imag;
        for(i = j; i < N; i += 1)
        {
                for(k = 2; k < 4; k += 1)
                {
                        (temp[i][k]).real = (F[i][2]).real*(matrix[0][k]).real - (F[i][2]).imag*(matrix[0][k]).imag
+ (F[i][3]).real*(matrix[1][k]).real - (F[i][3]).imag*(matrix[1][k]).imag;
                        (temp[i][k]).imag = (F[i][2]).imag*(matrix[0][k]).real +
(F[i][2]).real*(matrix[0][k]).imag + (F[i][3]).imag*(matrix[1][k]).real + (F[i][3]).real*(matrix[1][k]).imag;
                        //temp[i][k] = F[i][2]*matrix[0][k] - F[i+1][2]*matrix[1][k] + F[i][3]*matrix[2][k] -
F[i+1][3]*matrix[3][k];
                        //temp[i+1][k] = F[i+1][2]*matrix[0][k] + F[i][2]*matrix[1][k] + F[i+1][3]*matrix[2][k]
+ F[i][3]*matrix[3][k];
                }
        }
        for(i = j; i < N; i += 1)
        {
                for(k = 2; k < 4; k += 1)
                {
                        (F[i][k]).real = (temp[i][k]).real;
                        (F[i][k]).imag = (temp[i][k]).imag;
                }
        }

        // ... hyperbolic transform

        //r = sqrt(r1 + r2)*sqrt(r1 - r2);
        if(r1 < r2)
        {
                r.real = 0;
                r.imag = sqrt(r1 + r2)*sqrt(r2 - r1);
        }
        else
        {
                r.real = sqrt(r1 + r2)*sqrt(r1 - r2);
                r.imag = 0;
        }

        //c = r1/r;
        //s = r2/r;
        c.real = r1*r.real/(r.real*r.real + r.imag*r.imag);
        c.imag = -r1*r.imag/(r.real*r.real + r.imag*r.imag);
        s.real = r2*r.real/(r.real*r.real + r.imag*r.imag);
        s.imag = -r2*r.imag/(r.real*r.real + r.imag*r.imag);

        //F(index,[1 3) = F(index,[1 3])*[c', -s; s', c];
        (matrix[0][0]).real = c.real;
        (matrix[0][0]).imag = -c.imag;
```

```
                (matrix[0][2]).real = -s.real;
                (matrix[0][2]).imag = -s.imag;
                (matrix[1][0]).real = -s.real;
                (matrix[1][0]).imag = s.imag;
                (matrix[1][2]).real = c.real;
                (matrix[1][2]).imag = c.imag;
                for(i = j; i < N; i += 1)
                {
                        for(k = 0; k < 3; k += 2)
                        {
                                (temp[i][k]).real = (F[i][0]).real*(matrix[0][k]).real - (F[i][0]).imag*(matrix[0][k]).imag
+ (F[i][2]).real*(matrix[1][k]).real - (F[i][2]).imag*(matrix[1][k]).imag;
                                (temp[i][k]).imag = (F[i][0]).imag*(matrix[0][k]).real +
(F[i][0]).real*(matrix[0][k]).imag + (F[i][2]).imag*(matrix[1][k]).real + (F[i][2]).real*(matrix[1][k]).imag;
                                //temp[i][k] = F[i][0]*matrix[0][k] - F[i+1][0]*matrix[1][k] + F[i][2]*matrix[2][k] -
F[i+1][2]*matrix[3][k];
                                //temp[i+1][k] = F[i+1][0]*matrix[0][k] + F[i][0]*matrix[1][k] + F[i+1][2]*matrix[2][k]
+ F[i][2]*matrix[3][k];
                        }
                }
                for(i = j; i < N; i += 1)
                {
                        for(k = 0; k < 3; k += 2)
                        {
                                (F[i][k]).real = (temp[i][k]).real;
                                (F[i][k]).imag = (temp[i][k]).imag;
                        }
                }

                //L(index,j) = F(index,1);        % the j-th column of L
                for(i = j; i < N; i += 1)
                {
                        (L[i][j]).real = (F[i][0]).real;
                        (L[i][j]).imag = (F[i][0]).imag;
                }

                //L(j,j) = r;
                (L[j][j]).real = r.real;
                (L[j][j]).imag = r.imag;
        }
        return;
}
/****************************************************************
data_newPNcode.c
ADAPTIVE EQUALIZATION
Matlab code: James Thornbrue, June 6, 2001
C code: David Zaugg, Nov 1, 2001
------------------------------

Loads 10 channels into variable EQ_data
Channel data is length 999
Data provided by Mary Cassabaum
Requires files PNcodefilelist.txt, data_newPNcode1.bin, data_newPNcode2.bin
data_newPNcode3.bin, data_newPNcode4.bin, data_newPNcode5.bin
data_newPNcode6.bin, data_newPNcode7.bin, data_newPNcode8.bin
data_newPNcode9.bin, data_newPNcode10.bin
The data is written in binary (double) format in these files.
The data is preceded by four longs:
cols
rows
frames
bpp
real(data(1))
imag(data(1))
real(data(2))
imag(data(2))
...
real(data(end))
```

```
imag(data(end))
cols is 999, rows is 1, frames is 1, and bpp is 64 bits (double)
******************************************************************/

#include "new_approach.h"
#define LINELEN 80

struct COMPLEX* data_newPNcode(char inputfile[], struct COMPLEX* data, long *matrix_columns, long *matrix_rows)
{
        FILE *fid_filenames;
        FILE *fid_datafile;
        char filename[LINELEN];
        int N_files = 0;
        int NN, n, i;
        long cols = 0;
        long rows = 0;
        long frames = 0;
        long bpp = 0;
        int bytes;

        //open file with names of data files
        //each file contains a column of the
        //data matrix
        fid_filenames = fopen(inputfile, "r");
        if(fid_filenames == NULL)
        {
                printf("Error opening file with filenames.");
                exit(1);
        }

        //find out how many files there are (this is the
        //number of columns in the data matrix
        while(fgets(filename, LINELEN, fid_filenames) != NULL)
                N_files += 1;
        fseek(fid_filenames, 0, SEEK_SET);

        //find out number of cols and rows in one file
        //(one row in this case)
        fgets(filename, LINELEN, fid_filenames);
        NN = strlen(filename);
        for(n = 0; n < NN; n += 1)
        {
                if(filename[n] == '\n' || filename[n] == '\r')
                        filename[n] = NULL;
        }
        fid_datafile = fopen(filename, "rb");
        fread(&cols, 4, 1, fid_datafile);
        fread(&rows, 4, 1, fid_datafile);
        fread(&frames, 4, 1, fid_datafile);
        fread(&bpp, 4, 1, fid_datafile);

        //allocate space for the data matrix
        data = (struct COMPLEX*)malloc(bpp/8*2*cols*rows*N_files);
        if(data == NULL)
        {
                printf("Error: malloc failed to allocate memory for data matrix");
                exit(1);
        }
        fseek(fid_filenames, 0, SEEK_SET);

        //read data from files into data matrix
        for(i = 0; i < N_files; i += 1)
        {
                fgets(filename, LINELEN, fid_filenames);
                NN = strlen(filename);
                for(n = 0; n < NN; n += 1)
                {
                        if(filename[n] == '\n' || filename[n] == '\r')
```

```
                filename[n] = NULL;
            }
            fid_datafile = fopen(filename, "rb");
            fseek(fid_datafile, 16, SEEK_SET);
            for(n = 0; n < cols; n += 1)
            {
                bytes = fread(&((data[n + cols*i]).real), bpp/8, 1, fid_datafile);
                bytes = fread(&((data[n + cols*i]).imag), bpp/8, 1, fid_datafile);
                //conjugate
                (data[n + cols*i]).imag = -1.0*(data[n + cols*i]).imag;
                //origin + (0 for real, 1 for imag) + row index + column index
                if(bytes != 1)
                    printf("didn't read 8 bytes");
            }
            fclose(fid_datafile);
        }
        fclose(fid_filenames);
        *matrix_columns = cols;
        *matrix_rows = N_files;
        return data;
}
/*************************************************************
new_approach.h
HEADER FILE FOR NEW_APPROACH C PROGRAM
David Zaugg
Nov 1, 2001

Verify Xiaobai's Cholesky Decomposition Algorithm using real data.
Matlab code: James Thornbrue, June 18, 2001
C code: David Zaugg, Nov 1, 2001
*************************************************************/

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

//used in the fft
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

//N is size of covariance matrix, it must be 2^n long
#define N 16
//number of columns in F
#define Q 4

struct COMPLEX
{
        double real, imag;
} typedef;

//Does the FFT or IFFT in place
//FFT: isign = 1;
//IFFT: isign = -1, divide by nn.
//
//Inputs:
//data                              data vector
//nn                                length of the complex vector of the form
//                                          {real[0], imag[0], real[1], imag[1], ...,
//                                          real[nn-1], imag[nn-1]}
//                                          data is really 2*nn long
//isign                             indicates FFT or IFFT, 1 or -1 respectively
void four1(double data[], unsigned long nn, int isign);

//Reads data from files and stores in EQ_data
//
//Inputs:
//filename                  file containing names of files with data
//data                              column scanned data matrix, all values are conjugated
```

```
//                                          when loaded from files
//                                          each file contains a column of the data matrix
//cols                              number of columns
//rows                              number of rows
//format is data = {conj(file1), conj(file2), ..., conj(file10)}
struct COMPLEX* data_newPNcode(char filename[], struct COMPLEX* data, long *cols, long *rows);

//Forms first column of covariance matrix using FFT
//
//N                                         size of covariance matrix
//len_y                             length of y
//y                                         one of the columns of the data matrix depending on channel
//                                          column vector
//ref                               first column of the data matrix
//                                          rhs of LS system
//
//Outputs:
//a                                         first column of covariance matrix A
//b                                         not used, rhs of the covariance system
void FormCovFFT(int len_y, struct COMPLEX* y, struct COMPLEX* c, struct COMPLEX a[], struct COMPLEX b[]);

//Form the displacement structure
//
//fcol                              first column/row of covariance matrix
//yhead and ytail       from the parts of the data matrix that do not contribute
//                                          to every member of the covariance matrix
//                                          the first and last n-1 elements of the data vector
//outputs:
//F                                         displacement columns
//                                          used by DispL02 to form the Cholesky of the covariance matrix
//Sigma                             displacement structure
//                                          used by DispL02 to form the Cholesky of the covariance matrix
//A = ZAZ' + F/Sigma F';
void DispRank(struct COMPLEX fcol[], struct COMPLEX yhead[], struct COMPLEX ytail[], struct COMPLEX F[][Q], int
Sigma[]);

//Derive the cholesky decomposition L from the displacement structure
//
//F                                         displacement columns
//                                          used by DispL02 to form the Cholesky of the covariance matrix
//Sigma                             displacement structure
//                                          used by DispL02 to form the Cholesky of the covariance matrix
//
//Outputs:
//L                                         Cholesky decomposition
void DispL02(struct COMPLEX F[][Q], int Sigma[], struct COMPLEX L[][N]);
```

Appendix B: BSA Algorithm, Bld_sim_mtx routine, FTN-Code

```
      subroutine bld_sim_mtx ( n_mix,
     &                 n_reg,
     &                 avg,
     &                 sig,
     &                 var,
     &                 wgt,
     &                 sim_mtx )
      implicit none
      integer*4    n_mix      ! currently, allowed to be up to 100
      integer*4    n_reg      ! currently, allowed to be up to 200
      real*8     avg( 100 )
      real*8     sig( 100 )
      real*8     var( 100 )
      real*8     wgt( 100, 200 )
      real*8     sim_mtx( 200, 200 )
      real*8     var_sum
      real*8     aux( 100, 100 )
      real*8     recip_root_two_pi
      parameter  ( recip_root_two_pi = 0.39894 22804 01432 67794 d0 )
```

```
      real*8      sum
      real*8      tol
      parameter  ( tol = 1.0 d-07 )
      integer*4   ind_i
      integer*4   ind_j
      integer*4   ind_m
      integer*4   ind_n
C*******************************************************************************
C
C   Perform a series of input argument checks.
C
C*******************************************************************************
      if ( n_mix .lt. 1 ) then
        write ( 11, 101 ) n_mix
101     format ( /,
     &        ' ERROR 101 in routine "bld_sim_mtx":', /,
     &        ' n_mix = ', i5,                  /,
     &        ' but it must be POSITIVE.',      /,
     &        ' CALL EXIT now.',                / )
        call exit
      endif
C*******************************************************************************
      if ( n_mix .gt. 100 ) then
        write ( 11, 102 ) n_mix
102     format ( /,
     &        ' ERROR 102 in routine "bld_sim_mtx":', /,
     &        ' n_mix = ', i5,                  /,
     &        ' but it must not exceed 100.',   /,
     &        ' CALL EXIT now.',                / )
        call exit
      endif
C*******************************************************************************
      if ( n_reg .lt. 1 ) then
        write ( 11, 103 ) n_reg
103     format ( /,
     &        ' ERROR 103 in routine "bld_sim_mtx":', /,
     &        ' n_reg = ', i5,                  /,
     &        ' but it must be POSITIVE.',      /,
     &        ' CALL EXIT now.',                / )
        call exit
      endif
C*******************************************************************************
      if ( n_reg .gt. 200 ) then
        write ( 11, 104 ) n_reg
104     format ( /,
     &        ' ERROR 104 in routine "bld_sim_mtx":', /,
     &        ' n_reg = ', i5,                  /,
     &        ' but it must not exceed 200.',   /,
     &        ' CALL EXIT now.',                / )
        call exit
      endif
C*******************************************************************************
      do ind_i = 1, n_mix
        if ( sig(ind_i) .le. 0.0d0 ) then
          write ( 11, 105 ) ind_i, sig(ind_i)
105       format ( /,
     &          ' ERROR 105 in routine "bld_sim_mtx":', /,
     &          ' sig(', i5, ') = ', 1pd20.13,    /,
     &          ' but it must be POSITIVE.',      /,
     &          ' CALL EXIT now.',                / )
          call exit
        endif
      enddo
C*******************************************************************************
      do ind_i = 1, n_reg
        do ind_j = 1, n_mix
          if ( wgt(ind_j,ind_i) .lt. 0.0d0 ) then
            write ( 11, 106 ) ind_j, ind_i, wgt(ind_j,ind_i)
```

```
106      format ( /,
  &         ' ERROR 106 in routine "bld_sim_mtx":',      /,
  &         ' wgt for mixture ', i5, ' and region ', i5,
  &         ' = ', 1pd20.13,                  /,
  &         ' but it must be NON-NEGATIVE.',         /,
  &         ' CALL EXIT now.',                     / )
       call exit
      endif
     enddo
    enddo
C*******************************************************************************
    do ind_i = 1, n_reg
     sum = 0.0d0
     do ind_j = 1, n_mix
      sum = sum + wgt(ind_j,ind_i)
     enddo
     if ( abs( sum - 1.0d0 ) .gt. tol ) then
      write ( 11, 107 ) ind_i, tol
107     format ( /,
  &         ' ERROR 107 in routine "bld_sim_mtx":',       /,
  &         ' sum of wgt values for region ', i5,          /,
  &         ' differs from unity by more than:  ', 1pd20.13, /,
  &         ' CALL EXIT now.',                       / )
      call exit
     endif
    enddo
C*******************************************************************************
C
C   compute variance values from sigma values
C
C*******************************************************************************
    do ind_i = 1, n_mix
     var(ind_i) = sig(ind_i)**2
    enddo
C*******************************************************************************
C
C   compute common factor of wgt i interacting with wgt j
C
C*******************************************************************************
    do ind_i = 1, n_mix
     do ind_j = 1, n_mix
      var_sum = var(ind_i) + var(ind_j)
      aux( ind_i, ind_j ) = recip_root_two_pi
  &                  / sqrt( var_sum )
  &                  * exp( -0.5d0 * ( avg(ind_i) - avg(ind_j) )**2
  &                          / var_sum )
     enddo
    enddo
C*******************************************************************************
C
C   compute the entries for the similarity matrix,
C   using the ISE (integrated square error) approach
C
C*******************************************************************************
    do ind_m = 1, n_reg
     do ind_n = 1, n_reg
      sum = 0.0d0
      do ind_i = 1, n_mix
       do ind_j = 1, n_mix
        sum = sum + ( wgt( ind_i, ind_m ) * wgt( ind_j, ind_m )
  &              + wgt( ind_i, ind_n ) * wgt( ind_j, ind_n )
  &         - 2.0d0 * wgt( ind_i, ind_m ) * wgt( ind_j, ind_n ) )
  &         * aux( ind_i, ind_j )
       enddo
      enddo
      sim_mtx( ind_m, ind_n ) = sum
     enddo
    enddo
```

```
      return
      end
```

Appendix C: BSA Algorithm, Cluster_alg, FTN-Code

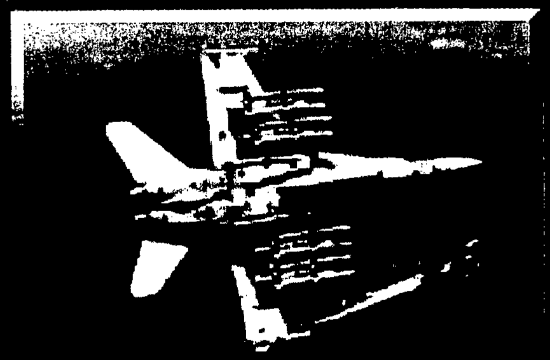
```
      Subroutine Cluster_alg ( Delta,
     &                         R_tilde,
     &                         D_matrix,
     &                         K_rho,
     &                         Gam,
     &                         mat_siz_max )
      Implicit none
      integer*4    mat_siz_max
      real*8       D_matrix( mat_siz_max, mat_siz_max )
      logical*1    K_rho   ( mat_siz_max, mat_siz_max )
      logical*1    T_rho   ( 100, 100 )
      logical*1    J_rho   ( 100, 100 )
      logical*1    T_tau   ( 100, 100 )
      integer*4    Card( 100 )
      real*8       Delta
      integer*4    R_tilde
      integer*4    Rho
      integer*4    Rho_prime
      integer*4    I_row
      integer*4    I_col
      integer*4    I_card
      integer*4    Loc_max
      integer*4    Val_max
      integer*4    Loc_count
      integer*4    Gam
      integer*4    Tau
      integer*4    Tau_prime
      logical*1    Duplicate
      logical*1    Aux_same
      logical*1    Empty
C***********************************************************************************
C
C   Perform a series of input argument checks.
C
C***********************************************************************************
      If ( Delta .le. 0.0d0 ) then
        Write ( 11, 101 ) Delta
101     Format ( /,
     &         ' ERROR 101 in routine "Cluster_alg":', /,
     &         ' Delta = ', 1pd20.13,               /,
     &         ' but it must be POSITIVE.',         /,
     &         ' CALL EXIT now.',                   / )
        Call exit
      Endif
C***********************************************************************************
      If ( R_tilde .lt. 1 ) then
        Write ( 11, 102 ) R_tilde
102     Format ( /,
     &         ' ERROR 102 in routine "Cluster_alg":', /,
     &         ' R_tilde =', i5,                    /,
     &         ' but it must be POSITIVE.',         /,
     &         ' CALL EXIT now.',                   / )
        Call exit
      Endif
C***********************************************************************************
      If ( R_tilde .gt. mat_siz_max ) then
        Write ( 11, 103 ) R_tilde, mat_siz_max
103     Format ( /,
     &         ' ERROR 103 in routine "Cluster_alg":',      /,
     &         ' R_tilde     =', i5,                 /,
     &         ' mat_siz_max =', i5,                 /,
     &         ' but the former must not exceed the latter.', /,
```

```
     &       ' CALL EXIT now.',                     / )
        Call exit
       Endif
C*******************************************************************************
       Do I_row = 1, R_tilde
        Do I_col = 1, R_tilde
         If ( I_row .lt. I_col ) then
          If ( D_matrix(I_row,I_col) .ne. D_matrix(I_col,I_row) ) then
           Write ( 11, 104 ) I_row, I_col, D_matrix(I_row,I_col),
     &                   I_col, I_row, D_matrix(I_col,I_row)
104         Format ( /,
     &             ' ERROR 104 in routine "Cluster_alg":',        /,
     &             ' D_matrix(', i5, ',' , i5, ') = ', 1pd20.13, /,
     &             ' D_matrix(', i5, ',' , i5, ') = ', 1pd20.13, /,
     &             ' but the matrix must be SYMMETRIC.',         /,
     &             ' CALL EXIT now.',                     / )
           Call exit
          Endif
         Endif
        Enddo
       Enddo
C*******************************************************************************
       Do I_row = 1, R_tilde
        If ( D_matrix(I_row,I_row) .ne. 0.0d0 ) then
         Write ( 11, 105 ) I_row, I_row, D_matrix(I_row,I_row)
105      Format ( /,
     &          ' ERROR 105 in routine "Cluster_alg":',      /,
     &          ' D_matrix(', i5, ',', i5, ') = ', 1pd20.13, /,
     &          ' but the diagonal elements must be ZERO.',  /,
     &          ' CALL EXIT now.',                    / )
         Call exit
        Endif
       Enddo
C*******************************************************************************
       Do I_row = 1, R_tilde
        Do I_col = 1, R_tilde
         If ( I_row .lt. I_col ) then
          If ( D_matrix(I_row,I_col) .le. 0.0d0 ) then
           Write ( 11, 106 ) I_row, I_col, D_matrix(I_row,I_col)
106         Format ( /,
     &             ' ERROR 106 in routine "Cluster_alg":',       /,
     &             ' D_matrix(', i5, ',', i5, ') = ', 1pd20.13, /,
     &             ' but off-diagonal terms must be POSITIVE.', /,
     &             ' CALL EXIT now.',                    / )
           Call exit
          Endif
         Endif
        Enddo
       Enddo
C*******************************************************************************
C
C     Done with the series of input argument checks.
C
C*******************************************************************************
       Do Tau = 1, R_tilde
        Do Tau_prime = 1, R_tilde
         T_tau( Tau, Tau_prime ) =
     &    D_matrix( Tau, Tau_prime ) .lt. Delta
         write ( 11, 999 ) Tau, Tau_prime, T_tau(Tau,Tau_prime)
999       format ( ' Tau, Tau_prime, T_tau(Tau,Tau_prime) = ',
     &          2i5, 4x, L1 )
        Enddo
       Enddo
c     call exit
C*******************************************************************************
       Rho_prime = 1
       Do I_col = 1, R_tilde
        T_rho( 1, I_col ) = T_tau( 1, I_col )
```

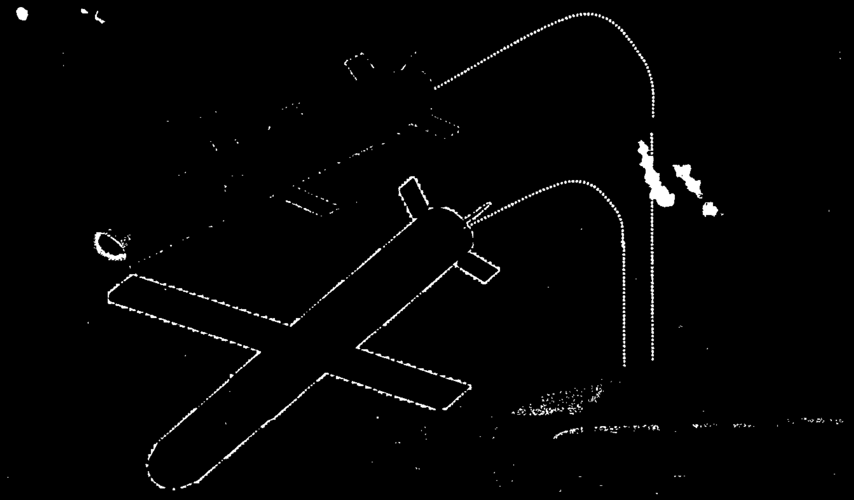
```
      Enddo
      Do Tau = 2, R_tilde
       Duplicate = .FALSE.
       Do Rho = 1, Rho_prime
        If ( .not. Duplicate ) then
         Aux_same = .TRUE.
         Do I_col = 1, R_tilde
          If ( T_tau( Tau, I_col ) .ne. T_rho( Rho, I_col ) ) then
           Aux_same = .FALSE.
          Endif
         Enddo
         If ( Aux_same ) then
          Duplicate = .TRUE.
         Endif
        Endif
       Enddo
       If ( .not. Duplicate ) then
        Rho_prime = Rho_prime + 1
        Do I_col = 1, R_tilde
         T_rho( Rho_prime, I_col ) = T_tau( Tau, I_col )
        Enddo
       Endif
      Enddo
      write ( 11, 998 ) Rho_prime
998   format ( /, ' Rho_prime = ', i5 )
c     call exit
C********************************************************************************
      Do Rho = 1, Rho_prime
       I_card = 0
       Do I_col = 1, R_tilde
        If ( T_rho( Rho, I_col ) ) then
         I_card = I_card + 1
        Endif
       Enddo
       Card(Rho) = I_card
      Enddo
C********************************************************************************
      Do Rho = 1, Rho_prime
       Do I_col = 1, R_tilde
        J_rho( Rho, I_col ) = .FALSE.
       Enddo
      Enddo
C********************************************************************************
      Do Tau = 1, R_tilde
       Loc_max = 0
       Val_max = 0
       Do Rho = 1, Rho_prime
        If ( T_rho( Rho, Tau ) ) then
         If ( Card(Rho) .gt. Val_max ) then
          Val_max = Card(Rho)
          Loc_max = Rho
         Endif
        Endif
       Enddo
       If ( Loc_max .lt. 1 ) then
        Write ( 11, 107 ) Tau, Loc_max
107     Format ( /,
     &        ' ERROR 107 in routine "Cluster_alg":', /,
     &        ' Tau     =', i5,                 /,
     &        ' Loc_max =', i5,                 /,
     &        ' but Loc_max must be POSITIVE.',       /,
     &        ' CALL EXIT now.',                  / )
        Call exit
       Endif
       If ( Loc_max .gt. Rho_prime ) then
        Write ( 11, 108 ) Tau, Rho_prime, Loc_max
108     Format ( /,
     &        ' ERROR 108 in routine "Cluster_alg":',   /,
```

```
   &       ' Tau      =', i5,                 /,
   &       ' Rho_prime =', i5,                 /,
   &       ' Loc_max   =', i5,                 /,
   &       ' but Loc_max must not exceed Rho_prime.', /,
   &       ' CALL EXIT now.',                  / )
       Call exit
     Endif
     J_rho( Loc_max, Tau ) = .TRUE.
   Enddo
C*******************************************************************************
   Do Tau = 1, R_tilde
     Loc_count = 0
     Do Rho = 1, Rho_prime
       If ( J_rho( Rho, Tau ) ) then
         Loc_count = Loc_count + 1
       Endif
     Enddo
     If ( Loc_count .ne. 1 ) then
       Write ( 11, 109 ) Tau, Loc_count
109    Format ( /,
   &       ' ERROR 109 in routine "Cluster_arg":', /,
   &       ' Tau      =', i5,                /,
   &       ' Loc_count =', i5,               /,
   &       ' but Loc_count must equal ONE.',      /,
   &       ' CALL EXIT now.',                / )
       Call exit
     Endif
   Enddo
C*******************************************************************************
   Gam = 0
   Do Rho = 1, Rho_prime
     Empty = .TRUE.
     Do Tau = 1, R_tilde
       If ( Empty ) then
         If ( J_rho( Rho, Tau ) ) then
           Empty = .FALSE.
         Endif
       Endif
     Enddo
     If ( .not. Empty ) then
       Gam = Gam + 1
       Do Tau = 1, R_tilde
         K_rho( Gam, Tau ) = J_rho( Rho, Tau )
       Enddo
     Endif
   Enddo
C*******************************************************************************
   Do I_row = 1, Gam
     Write ( 11, 201 ) I_row
201  Format ( /,
   &       ' Segment Region Number:', i5, / )
     Do I_col = 1, R_tilde
       If ( K_rho( I_row, I_col ) ) then
         Write ( 11, 202 ) I_col
202        Format ( '    contains subregion:', i5 )
       Endif
     Enddo
   Enddo
C*******************************************************************************
   Return
   End
```

ADVANCED MATHEMATICS FOR MISSILE SEEKER SIGNAL PROCESSING



AMRAAM



NETFIRES PAM