

AD-A155 932

MULTIPLE SENSOR TRACKING IN THE INTERIM BATTLE GROUP
TACTICAL TRAINER(U) NAVAL POSTGRADUATE SCHOOL MONTEREY
CA K W SPANGENBERG MAR 85

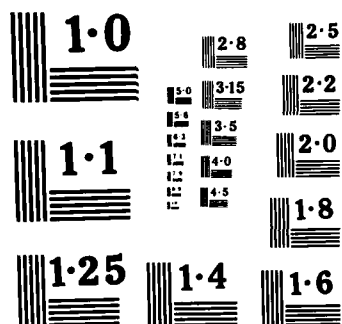
1/1

UNCLASSIFIED

F/G 17/1

NL

								END					



NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

2

AD-A155 932

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
JUL 3 1985
B

THESIS

MULTIPLE SENSOR TRACKING IN THE
INTERIM BATTLE GROUP TACTICAL TRAINER

by

Keith N. Spangenberg
March 1985

Thesis Advisor:

Rex H. Shudde

Approved for public release; distribution is unlimited

DTIC FILE COPY

RE P 00 000

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
	AD-A155	932
4. TITLE (and Subtitle) Multiple Sensor Tracking in the Interim Battle Group Tactical Trainer		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1985
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Keith N. Spangenberg		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE March 1985
		13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution is unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Kalman Filter, Tracking Model, Interim Battle Group Tactical Trainer, Naval Warfare Interactive Simulation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The thesis provides two subroutines for the Interim Battle Group Tactical Trainer (IBGTT). The first subroutine is a single sensor tracking model using the Kalman filter. This subroutine is part of the Passive Sonar Model. The second subroutine is a multiple sensor tracking model using the Kalman filter to correlate all sensor contacts on a specific target. This subroutine is a separate entity and can be turned		

20. Continued

on or off at simulation initiation as required by training objectives.

Precedence For	
1. COMBAT	☒
2. GAD	☐
3. Unassigned	☐
Simulation	
Distribution/	
Availability Codes	
and/or	
Last	
A-1	

Approved for public release; distribution is unlimited.

Multiple Sensor Tracking
in the
Interim Battle Group Tactical Trainer

by

Keith N. Spangenberg
Lieutenant, United States Navy
B.S., United States Naval Academy, 1977

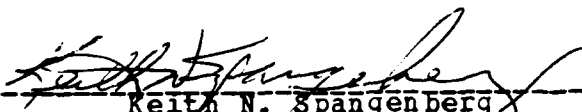
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEM TECHNOLOGY
(ANTISUBMARINE WARFARE)

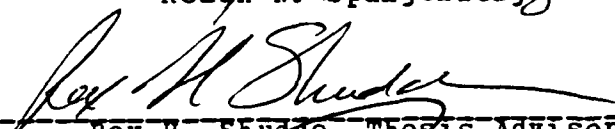
from the

NAVAL POSTGRADUATE SCHOOL
March 1985

Author:


Keith N. Spangenberg

Approved by:


Rex H. Shuade, Thesis Advisor


R. Neagle Forrest, Second Reader


R. Neagle Forrest, Chairman,
Antisubmarine Warfare Academic Group


David A. Schrad,
Academic Dean

ABSTRACT

The thesis provides two subroutines for the Interim Battle Group Tactical Trainer (IBGTT). The first subroutine is a single sensor tracking model using the Kalman filter. This subroutine is part of the Passive Sonar Model. The second subroutine is a multiple sensor tracking model using the Kalman filter to correlate all sonar contacts on a specific target. This subroutine is a separate entity and can be turned on or off at simulation initiation as required by training objectives.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CCNTENTS

I.	INTRODUCTION	8
II.	SINGLE SENSOR TRACKING	11
	A. TARGET MOTION ANALYSIS (TMA) MODEL	11
	B. KALMAN FILTER IMPROVEMENTS	14
	C. KALMAN FILTER MODEL	15
	D. TRACK QUALITY	18
	E. CHANGES TO OTHER SUBROUTINES	19
	1. Subroutine WARCYC	19
	2. Subroutine LCLPSN	20
III.	MULTIPLE SENSOR TRACKING	21
	A. SONAR CORRELATION MODEL	21
	B. KALMAN FILTER REPLACEMENT	21
	C. KALMAN FILTER MODEL	22
	D. CHANGES TO OTHER SUBROUTINES	23
IV.	TEST RESULTS	24
	APPENDIX A: SINGLE SENSOR MODEL (RATIONAL FORTRAN)	25
	APPENDIX B: SINGLE SENSOR MODEL (SOURCE CODE)	31
	APPENDIX C: MULTIPLE SENSOR MOEL (RATIONAL FORTRAN)	37
	APPENDIX D: MULTIPLE SENSOR MOEL (SOURCE CODE)	48
	BIBLIOGRAPHY	60
	INITIAL DISTRIBUTION LIST	61

LIST OF TABLES

I	TMA Error Parameters	12
II	TMA Quality	13

I. INTRODUCTION

The Naval Ocean Systems Center has developed the Interim Battle Group Tactical Trainer/Computer Support Facility (IBGTT/CSF) as a computer-based tactical simulation system to provide a training device for senior naval officers to practice tactical decision making until such time as the Enhanced Naval Warfare Gaming System becomes available. The trainer is intended to provide interactive, multithreat, multiplatform operational situations in a simulated yet realistic operational environment so that selected officers can study, practice, and be evaluated in force-level tactical decision making.

The IBGTT training capability is implemented as a real-time, man-interactive, computer-aided (discrete event, time step) simulation of the naval warfare environment. In operation, the IBGTT supports a two-sided (BLUE vs. ORANGE) interactive scenario in which opposing sides can define, structure, and dynamically control forces ranging in size from one or more battle groups and associated aircraft, down to a single air or surface unit. Force elements and their associated characteristics, sensors, weapons, and communication systems may be derived from real, proposed, or conceptualized units or systems.

The utilization of IBGTT involves the use of the four major Naval Warfare Interactive Simulation System (NWISS) processes; BUILD, FORCE, WARGAME, and POST-GAME ANALYSIS. The BUILD process is a stand alone interactive program used to create and maintain platform, sensor, communication, and weapon characteristics in the IBGTT Characteristic Data Base. The FORCE process is a stand alone interactive program used to create and maintain an exercise scenario using the

data base created by the BUILD process. The WARGAME process is an interactive program used to accept and execute user orders; control platform motion, detections, and communications; determine engagement and other event outcomes; and display status information and tactical situations. The POST-GAME ANALYSIS process analyzes and lists critical data recorded during the exercise; supports exercise reconstruction; and tentatively evaluates some Measures of Effectiveness.

A global data area in NWISS, called the blackboard, is shared by all the major modules functioning during the exercise. It is the area where these modules interface with each other through the application of uniform naming conventions and the efficient use of memory. The blackboard is essentially comprised of numerous tables and subtables. Each table is assigned specific pointers while each subtable is assigned specific indices. The tables and subtables contain the fields (data) that are unique to that particular table or subtable. Each data item in the blackboard is referred to as a field and includes both whole words and specific bits. The field names are structured to provide the identity of the associated pointers and indices as well as the data type in the field.

Effective training at this level requires models of naval warfare interactions which provide realistic results based on an emulation of the actual warfare system. NWISS uses a wide variety of models to simulate the behavior of platforms, weapons, sensors, and communication systems. However, many of these models do not emulate the actual warfare system nor do they provide realistic results. Therefore, it is necessary to improve or replace these deficient models in order to obtain effective training and meet the objectives for which IBGTT was designed.

This thesis will address two models in particular. The first model is the Target Motion Analysis (TMA) Model which processes passive sonar contacts. The current model will be examined, followed by a presentation of a Kalman filter improvement to the model. Secondly, the Sonar Correlation Model will be examined, followed by a presentation of a Kalman filter to replace the current model.

The reader should be advised that the TMA Kalman Filter Model is in actuality equivalent to the Sonar Correlation Kalman Filter Model, as will become evident in the presentation of the two Kalman Filter Models. This thesis further presupposes that the reader is familiar with the Kalman filter.

II. SINGLE SENSOR TRACKING

A. TARGET MOTION ANALYSIS (TMA) MODEL

The current model will monitor the number of game minutes for which passive contact has been held on each target by each detecting passive sonar (i.e., submarine, surface sonar, towed array, or sonobuoy). When this time exceeds the TMA time defined by the user at simulation initiation a target motion analysis report will be displayed on the Passive Sonar Status Board. The TMA range, course, and speed displayed are derived as follows:

$$\text{TMA} \begin{bmatrix} \text{range} \\ \text{course} \\ \text{speed} \end{bmatrix} = \text{Actual target} \begin{bmatrix} \text{range} \\ \text{course} \\ \text{speed} \end{bmatrix} \pm \text{FACTOR}$$

The FACTOR is the product of a random number drawn from a uniform distribution and a derived parameter.

These parameters, which are indicated in Table I, cause increasingly accurate solutions to be developed as Signal Excess (SE) increases and as the target's true bearing changes ($\bullet B = \Delta B$) from the true bearing of its initial detection. This latter factor simulates improved solutions derived from higher bearing rate targets and longer tracking times. The solution quality displayed will be selected from Table II as a function of SE and $\bullet B$.

Once a TMA solution has been displayed, only the range is updated on the display. The range only update continues until the signal excess and/or change in true bearing cause a new parameter to be developed from the table (e.g., SE changes from -6 to -5 or $\bullet B$ changes from 5 to 6). When a new parameter is selected, a new TMA range, course, and quality

SINGLE SENSOR MODEL (RATIONAL FORTRAN)

25

IV. TEST RESULTS

The single sensor and multiple sensor models have been tested on a limited basis. Each subroutine has been tested independently to insure that the models perform as designed. However, the subroutines have not been tested for integration into the overall NWISS program or other subroutines.

Due to time constraints and computer availability, integration tests were not possible. The added blackboard space required has not been made but should not pose any problems. Inherent with any new subroutine is the unforeseeable affect it may have on unrelated subroutines. This aspect of testing has not been performed.

The track quality will be the same as that presented in Chapter Two.

D. CHANGES TO OTHER SUBROUTINES

The rational FORTRAN and source code listing for this model are contained in Appendix C and D, respectively. The changes presented in Chapter Two for Subroutine WARCYC are also applicable to this model. The only other change that will be necessary is that all active information needs to be added to the Remote Table.

no fix or track is associated with the correlation. As a result, each game minute two bearing lines are displayed (they may not be the same two from the previous minute nor necessarily an improvement) that jump around the screen, providing no useful information to the user.

The Kalman Filter Model permits all the information available, both active and passive, to be correlated on a specific target and be displayed as a fix with an updated track. In addition, the track quality associated with the fix provides the user with the added information about the relative size of the AOP. The on and off ability of the Sonar Correlation Model will be incorporated into the Kalman Filter Model for the previously stated reasons concerning its flexibility.

C. KALMAN FILTER MODEL

As mentioned in the introduction, the single sensor tracking model is in actuality equivalent to the multiple sensor tracking model. The difference being the scope of the information being processed. The single sensor model is a subset of the Passive Sonar Model; whereas, the Multiple sensor model is a separate entity correlating all available information. The basics of the models, including assumptions and initial conditions, are the same. Therefore, only the differences from the single sensor tracking model, presented in Chapter Two, will be discussed here.

The model will handle bearing and range measurements as well as bearing only measurements. The bearings are ± 5 degrees and the ranges are ± 5 nautical miles. For the bearing only measurement, the H matrix will be the same as for the single sensor tracking model. For the bearing and range measurement, the following H matrix will be utilized:

$$H = \begin{bmatrix} -\sin(\theta)/range & \cos(\theta)/range & 0 & 0 \\ \cos(\theta) & \sin(\theta) & 0 & 0 \end{bmatrix}$$

III. MULTIPLE SENSOR TRACKING

A. SONAR CORRELATION MODEL

The Sonar Correlation Model would be more appropriately described as a procedure instead of a model. This routine determines the correlation of bearings between two detecting units at a specific target and performs two functions. First, the two detecting units (with an intersection angle of at least 60 degrees, or else the largest available angle) to a common target will display bearing lines. Second, multiple targets detected within a certain maximum arc will display only one line; and will set the composition field set (i.e., one, few, or many). All other passive sonar lines will not be displayed.

The Sonar Correlation Model can be turned on or off. This allows for the flexibility of being utilized for Battle Group Commanders and their staff when the "big picture" is the main concern and not the individual unit prosecution; and yet, be turned off when the trainer is being utilized for a single unit or group of units practicing coordinated operations.

B. KALMAN FILTER REPLACEMENT

The current procedure has many drawbacks. Only passive sonar lines are considered; active information is displayed separately and is not correlated with the passive information.

The routine searches through the Remote Table until it finds two bearing lines that meet the 60 degree criteria. These are displayed and the routine ceases to search; thus, not necessarily choosing the optimum solution. In addition,

2. Subroutine LCLPSN

- Change line 9 to read:

```
EQUIVALENCE (IBB,FBB,CBB,IBBW,IBBB,PBB)
```

This includes the blackboard of the P matrix.

- After line 113 add:

```
I_LCL$LASTTMA TIME=(IAND(ISHFT(IBB(KPOINT_LCL  
*+1),-0),65535)
```

This time is used in Subroutine LCLTMA.

- Change line 262 to read:

```
IF(.NOT.(I_LCL$OMNIFLAG.NE.1)) GOTO 23269
```

This removes the TMA exceed time criteria.

- Change line 264 to read:

```
LCLTMA(KPOINT_LCL,RORLAT,RORLON,LBEAR,  
*I_LCL$PMATRIX,I_LCL$LASTTMA TIME)
```

- If the 2-sigma axis is less than or equal to 500 yards, then the track receives a quality of GOOD. The criteria of 500 yards is used because the Engagement Model employs a 500 yard kill radius for a torpedo.
- If the 2-sigma axis is greater than 500 yards but less than or equal to 1000 yards, then the track receives a quality of FAIR. The criteria of 1000 yards was chosen simply because it is twice the GOOD criteria.
- If the 2-sigma axis is greater than 1000 yards, then the track receives a quality of POOR.

E. CHANGES TO OTHER SUBROUTINES

Implementation of this subroutine (LCLTMA, rational FORTRAN and source code listing are contained in Appendix A and B, respectively) will require some changes to other subroutines and additions to the blackboard. These changes and/or additions include:

1. Subroutine WARCYC

Each target needs a $P(0)$ matrix in the blackboard; therefore, add:

```

      REAL PBB(4,4)
      REAL I_LCL$PMATRIX(4,4), I_RMT$PMATRIX(4,4)
      DO 50000 J=1,4
      DO 50001 K=1,4
      IF(.NOT.(J.EQ.K))GOTO 50001
      PBB(J,K) = 1000.
      GOTO 50000
50001 PBB(J,K) = 0.
50000 CONTINUE
      I_LCL$PMATRIX = PBB
      I_RMT$PMATRIX = PBB

```

θ_0 is the initial observation

and

$$P(0) = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix}$$

Note: 1000 was chosen since it is approximately equal to $\pm$ one convergence zone (32 nm) and $\pm$ 32 knots.

The first assumption is that

$$E[W(k) * V(j)^{\text{transpose}}] = 0 \text{ for all } j \text{ and } k.$$

This means that the plant noise and the measurement noise are uncorrelated. Secondly, recall that the assumption is that during an encounter, the target's course and speed remain constant.

D. TRACK QUALITY

In order to reduce the number of changes required to the overall program, the TMA quality currently used from Table II will be utilized but based on different criteria. This will eliminate the need to change the blackboard; and more importantly, will not change the Status Tableau seen by the player, which is already full.

The track quality is based on the semi-major axis of the AOP. This is determined from the error covariance matrix as follows:

$$(\text{semi-major axis})^2 = (p_{11} + p_{22}) / 2 + \text{SQRT}[(p_{11} - p_{22})^2 / 4 + p_{12}^2]$$

where p_{ij} are the elements of the $P(t)$ matrix. A 2-sigma semi-major axis is used to insure a probability of 0.8647. The track quality is then determined as follows:

$V(t)$ is the measurement noise. It is approximately $N(0, R(t))$. For this model, all bearings are $\pm .5$ degrees.

$K(t)$ is the Kalman gain. The update from $\hat{X}(t)$ just before the measurement to $\hat{X}(t)$ just after the measurement is proportional to the shock; the Kalman gain is the proportionality constant.

It was stated earlier that the measurements are assumed to be linearly related to the system state. Since the measurement is in polar coordinates, $h(x)$ is in fact nonlinear. Therefore, a transformation must be made on $h(x)$ to give a linearly related H .

In this model, the observation will be made from a platform at (u, w) to a target at (x, y) , where x is north and y is east. So,

$$h(X) = \theta = \tan^{-1}[(y-w)/(x-u)]$$

or,

$$H = [\partial h(X)/\partial x \quad \partial h(X)/\partial y \quad \partial h(X)/\partial x' \quad \partial h(X)/\partial y']$$

∂ represents the partial derivative

evaluated at $X = \hat{X}$. Hence,

$$H = [-\sin(\theta)/range \quad \cos(\theta)/range \quad 0 \quad 0]$$

This model was built upon two initial conditions and two important assumptions. The initial conditions are:

$$E[X(0)] = \hat{X}(0)$$

and

$$E\{[X(0) - \hat{X}(0)] * [X(0) - \hat{X}(0)]^T\} = P(0)$$

where

$$\hat{X}(0) = [32\cos\theta_0 \quad 32\sin\theta_0 \quad 0 \quad 0]^T$$

P is the error covariance matrix.

$$P(t) = E[\{X(t) - \hat{X}(t)\} \{X(t) - \hat{X}(t)\}^T]$$

W(t) is the plant noise. It describes the randomness of the system as it moves from state X(t) to X(t+1). W(t) is approximately N(0, Q(t)). For this model, Q is taken to be zero.

Next, a new fix is computed based on an observation. This is determined from the measurement model:

$$Z(t) = H(t) * X(t) + V(t)$$

Thus, measurement is:

Kalman Gain: K(t) =

$$P(t) * H(t)^T * [H(t) * P(t) * H(t)^T + R(t)]^{-1},$$

State Update:

$$\hat{X}(t) := \hat{X}(t) + K(t) * [Z(t) - H(t) * \hat{X}(t)],$$

and

Error Covariance Update:

$$P(t) := P(t) - K(t) * [P(t) * H(t)^T] * K(t)^T$$

where

:= indicates that the right hand side is computed and replaces the value on the left hand side of the symbol.

Z(t) is the actual measurement. The measurements are assumed to be linearly related to the system state X(t) by the observation matrix H(t). Note: H(t) * $\hat{X}(t)$ is the predicted outcome of the measurement. The difference, Z(t) - H(t) * $\hat{X}(t)$, is the measurement residual or shock.

problem since the computed AOF updates smoothly with the sensor information.

C. KALMAN FILTER MODEL

First of all, it is assumed that during an encounter, the target's course and speed remain constant. The model updates the position of the fix since the last observation based on the previous estimate. This is based on the system model:

$$X(t) = \text{PHI}(t-1) * X(t-1) + W(t-1)$$

Thus, movement is:

$$\text{State Extrapolation: } \hat{X}(t) = \text{PHI}(t-1) * \hat{X}(t-1)$$

and

Error Covariance Extrapolation:

$$P(t) = \text{PHI}(t-1) * P(t-1) * \text{PHI}^T(t-1) + Q(t-1)$$

where

$\hat{X}$ is the estimated state vector. It is assumed to be multivariate normal with mean zero.

$$X(t) = [x(t) \quad y(t) \quad x' \quad y']^T$$

$x' = x \text{ velocity}$
 $y' = y \text{ velocity}$

PHI is the transition matrix. It describes how the state vector changes from $X(t)$ to $X(t+1)$.

$$\text{PHI} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \Delta t = \text{delta } t$$

B. KALMAN FILTER IMPROVEMENTS

The model does not begin to compute a track until the TMA time defined by the user at simulation initiation is exceeded. This implies that all sensors and operators are equal, which is not realistic. Furthermore, this does not allow for accurate information to be used at time of detection unless the TMA time has already been exceeded. For instance, a passive sonobuoy dropped in front of a contact, producing a CPA (closest point of approach) for its initial detection, would have good track information that would not be utilized by the model; for only information after the TMA time is used in determining the FACTOR.

This TMA initiation time was included in the model because use of actual target information provided too accurate of a fix, even with bad sensor information, for a player to experience a realistic prosecution. The Kalman Filter Model eliminates the need for this waiting time since the model receives information as an operator would see it (that is, apparent bearing resulting from apparent position, including navigation error, and sensor bearing error). Thus, the Kalman Filter Model allows use of all sensor information with appropriate errors to provide realistic prosecution.

The TMA model attempts to simulate a changing area of probability (AOP) with improved solutions taken from the table as SE increases and true bearing changes. The problem of using true information instead of apparent has already been discussed. In addition, the improved AOP is heavily dependent upon the drawing of a random number. It has been observed in actual trainers that the AOP fluctuates as randomly as the random number generator, regardless of sensor information; providing confusing information to the player. Again, the Kalman Filter Model eliminates this

$0 < SE$	$.2S$	$.1S$	$.05S$	0

R = Actual Range				
S = Actual Speed				

TABLE II					
TMA Quality					
SIGNAL EXCESS (SE) IN dB	BEARING CHANGE IN DEGREES (•E)	•B<5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12		POOR	POOR	FAIR	FAIR
-12 ≤ SE ≤ -6		POOR	FAIR	FAIR	FAIR
-6 ≤ SE ≤ 0		FAIR	FAIR	FAIR	GOOD
0 < SE		FAIR	FAIR	GOOD	GOOD

will be calculated and displayed. Between TMA changes, the displayed range is updated each simulation cycle to show the range of the target estimated from the TMA course and speed.

If contact is lost for a time greater than the user-defined track loss time, the TMA solution will no longer be displayed. At any subsequent redetection of the same target, a new solution will be generated after the appropriate time interval has passed.

TABLE I
TMA Error Parameters

RELATIVE RANGE ERROR					
SIGNAL EXCESS (SE) IN dB	BEARING CHANGE IN DEGREES (•B)	•B≤5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12		R	.8R	.7R	.4R
-12 < SE ≤ -6		.8R	.7R	.4R	.25R
-6 < SE ≤ 0		.7R	.4R	.25R	.1R
0 < SE		.4R	.2R	.1R	.05R
COURSE ERROR IN DEGREES					
SIGNAL EXCESS (SE) IN dB	BEARING CHANGE IN DEGREES (•B)	•B≤5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12		120	90	45	30
-12 < SE ≤ -6		90	45	30	15
-6 < SE ≤ 0		45	30	15	7
0 < SE		30	15	7	5
RELATIVE SPEED ERROR					
SIGNAL EXCESS (SE) IN dB	BEARING CHANGE IN DEGREES (•B)	•B≤5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12		.5S	.4S	.3S	.2S
-12 < SE ≤ -6		.4S	.3S	.2S	.1S
-6 ≤ SE ≤ 0		.3S	.2S	.1S	.05S

```

real  XEST(4),X(4)
      PHI(4,4)
      PHIT(4,4)
      P(4,4),I_LCL$PMATRIX(4,4)
      H(4)
      K(4)
      Z

      #System Model State Vector
      #System Model Transition Matrix
      #Transpose of Transition Matrix
      #Error Covariance Matrix
      #Measurement Model Observation Matrix
      #Kalman Gain
      #Measurement Vector

      #Estimated LAT/LON
      # of TGT
      #Estimated COURSE and
      # SPEED of TGT

      #State Extrapolation:
      #Distance in north-south direction
      #Insure shortway around earth
      #Distance in east-west direction
      #Speed vector in north-south direction
      #Speed vector in east-west direction

      #Initialize PHI Matrix
      PHI(1,3)=PHI(2,4)=$delta$time      #Movement time in hrs since last

      LCL00210
      LCL00220
      LCL00230
      LCL00240
      LCL00250
      LCL00260
      LCL00270
      LCL00280
      LCL00290
      LCL00300
      LCL00310
      LCL00320
      LCL00330
      LCL00340
      LCL00350
      LCL00360
      LCL00370
      LCL00380
      LCL00390
      LCL00400
      LCL00410
      LCL00420
      LCL00430
      LCL00440

```

#update	LCL00450
	LCL00460
#X\$hat=PHI*X\$hat	LCL00470
	LCL00480
#PHI\$transpose	LCL00490
#P=PHI*P*PHI\$transpose	LCL00500
	LCL00510
	LCL00520
	LCL00530
	LCL00540
	LCL00550
	LCL00560
	LCL00570
	LCL00580
	LCL00590
	LCL00600
	LCL00610
	LCL00620
	LCL00630
	LCL00640
	LCL00650
	LCL00660
	LCL00670
	LCL00680


```

#X$hat=PHI*X$hat
#State Extrapolation Vector

#PHI$transpose
#P=PHI*P*PHI$transpose
#Error Covariance Extrapolation

#MEASUREMENT:

#Estimated bearing
#Estimated range

#Measurement Observation Matrix
H (1) = -SIN (THETA$hat) / RNG$hat
H (2) = COS (THETA$hat) / RNG$hat
H (3) = H (4) = 0.

#P*H$transpose
#H*P*H$transpose
#H*P*H$transpose+R
#Kalman Gain
#H*X$hat
ZH$X=Z-HX
#K* (Z-H*X$hat)
= (P*H$transpose)$transpose
R=BRG measurement error=1.5 degrees
#Measurement Residual

```

```

#X$hat=X$hat+K*(Z-H*X$hat)      State Update      LCL00690
LCL00700
#K*(P*H$transpose)$transpose   LCL00710
#P=P-K*(P*H$transpose)$transpose #Error Covariance Update LCL00720
LCL00730
#New estimated bearing          LCL00740
#New estimated range           LCL00750
LCL00760
#Compute LAT/LON of ERG/RNG from ORIGIN LCL00770
CALL RRB2LL(_                   #Get LAT/LON      LCL00780
             F_LCL$TMALAT,      #ORIGIN LAT -> FIX LAT (input/output) LCL00790
             F_LCL$TMALON,      #ORIGIN LON -> FIX LON (input/output) LCL00800
             RNG,               #Range from ORIGIN to TGT      LCL00810
             THETA,             #Bearing from ORIGIN to TGT    LCL00820
             0.0,               #Pass zero                    LCL00830
             COSI)              #Cosine of LAT (input/output) LCL00840
LCL00850
putLCL$TMALAT$f                 #New FIX position      LCL00860
putLCL$TMALON$f                 LCL00870
LCL00880
#New estimated course          LCL00890
#New estimated speed           LCL00900

```

```
LCL00910
LCL00920      #New FIX course
LCL00930      #    and speed
LCL00940
LCL00950      #Determine semi-major axis of area of probability
LCL00960          #SIGMA$Squared=(P11+P22)/2+SQRT{((P11-P22)+(P11-P22))/4+P12*P12}
LCL00970          #2SIGMA=2*SIGMA/2025 yds
LCL00980
LCL00990
LCL01000          TMA$Quality=2          #GCOD
LCL01010      else if(2SIGMA > 500 yds & <= 1000 yds)then
LCL01020          TMA$Quality=1          #FAIR
LCL01030      else (2SIGMA > 1000 yds)
LCL01040          TMA$Quality=0          #PCOR
LCL01050
LCL01060
LCL01070      return
LCL01080      end          #End lclTMA
#####
Subroutine MUL4X4(A,
                  B,
                  C)
#####
      #4X4 matrix (input)
      #4X4 matrix (input)
      #4X4 matrix (output)
#####
#####LCL01120
LCL01130
```

```

# Purpose: Multiplies two 4X4 matrices together.
#
# Called by: LCLTMA  CORSNR
#
#####LCL01180
LCL01140
LCL01150
LCL01160
LCL01170
#####
LCL01190
LCL01200
LCL01210
LCL01220
LCL01230

C=A*B

return
end
#End MUL4X4

```

APPENDIX B
SINGLE SENSOR MODEL (SOURCE CODE)

```

SUBROUTINE LCLTMA(KPOINT_LCL,RORLAT,RORLON,LBEAR,I_LCL$PMATRIX,I_LLCL00010
*CL$LASTTIME)          LCL00020
IMPLICIT REAL*8 (A,C)   LCL00030
INTEGER IBB(1025),IBBP(6,85) LCL00040
INTEGER*2 IBBW(2,1025)  LCL00050
BYTE IBBB(4,1025)       LCL00060
REAL*8 CBB(512)         LCL00070
REAL FBB(1025),I_LCL$PMATRIX(4,4),P(4,4),KPHTT(4,4),H(4),PHT(4) LCL00080
REAL K(4),HPHT,HPHTR,HX,PROD(4),XEST(4),X(4),PHI(4,4),PHIT(4,4) LCL00090
REAL PBB(4,1025)        LCL00100
EQUIVALENCE (IEE,FBB,CBB,IBBW,IBBB,PBB) LCL00110
EQUIVALENCE (IBB(513),IBBE) LCL00120
COMMON/BBOARD/IEB       LCL00130
F_LCL$TMALAT=(IAND(ISHFT(IEB(KPOINT_LCL+8),-0),65535)*1.*.0001-3.2LCL00140
*)
F_LCL$TMALON=(IAND(ISHFT(IEB(KPOINT_LCL+8),-16),65535)*1.*.0001-3.LCL00160
*2)
F_LCL$TMACSE=(IAND(ISHFT(IEB(KPOINT_LCL+5),-0),511)) LCL00170
F_LCL$TMASPD=(IAND(ISHFT(IEB(KPOINT_LCL+4),-16),4095)) LCL00180
XEST(1)=F_LCL$TMALAT-RORLAT LCL00190
DELLON=F_LCL$TMALON-RORLON LCL00200
                                LCL00210

```

```

ANGPI (DELLON)
COST=COS (F_LCL$TMLAT)
COSL=COS (RORLAT)
XEST (2)=.5*(COSI+COST)*DELLON
XEST (3)=F_LCL$TMA SPD*COS (F_LCL$TMACSE)
XEST (4)=F_LCL$TMA SPD*SIN (F_LCL$TMACSE)
DO 50002 J=1,4
DO 50002 K=1,4
IF (.NOT. (J.EQ. K)) GOTO 50003
PHI (J,K)=1.
GOTO 50002
50003 PHI (J,K)=0.
50002 CONTINUE
PHI (1,3)=(IBB (103)-I_LCL$IASTTIME)/60.
PHI (2,4)=PHI (1,3)
DO 50004 J=1,4
X (J)=0.
DO 50004 K=1,4
50004 X (J)=X (J)+PHI (J,K)*XEST (K)
DO 50005 J=1,4
DO 50005 K=1,4
50005 PHIT (J,K)=PHI (K,J)
CALL MUL4X4 (PHI,I_LCL$PMATRIX,P)
CALL MUL4X4 (P,PHIT,I_LCL$PMATRIX)

```

```

      THETA=ATAN2 (X (2) ,X (1))
      THETA=INT (THETA*(180./3.141592654)+.5)
      RNG=SQR(X (1) *X (1)+X (2) *X (2))
      H (1)=-SIN (THETA)/RNG
      H (2)=COS (THETA) /RNG
      H (3)=0.
      H (4)=0.
      DO 50006 J=1,4
      PHT (J)=0.
      DO 50006 K=1,4
      PHT (J) =PHT (J) +I_LCL$PMATRIX (J,K) *H (K)
      HPHT=0.
      DO 50007 J=1,4
      50007 HPHT=HPHT+H (J) *PHT (J)
      HPHTR=HPHT+.25
      DO 50008 J=1,4
      50008 K (J) =PHT (J) /HPHTR
      Z=FLOAT (LBEAR)
      HX=0.
      DO 50009 J=1,4
      50009 HX=HX+H (J) *X (J)
      ZHX=Z-HX

```

LCL00460
 LCL00470
 LCL00480
 LCL00490
 LCL00500
 LCL00510
 LCL00520
 LCL00530
 LCL00540
 LCL00550
 LCL00560
 LCL00570
 LCL00580
 LCL00590
 LCL00600
 LCL00610
 LCL00620
 LCL00630
 LCL00640
 LCL00650
 LCL00660
 LCL00670

```

DO 50010 J=1,4
50010 PRCD(J)=K(J)*ZHX
DO 50011 J=1,4
50011 XEST(J)=X(J)+PROD(J)
DO 50012 J=1,4
DO 50012 L=1,4
50012 KPHTT(J,L)=K(J)*PHT(L)
DO 50013 J=1,4
DO 50013 K=1,4
50013 I_LCL$PMATRIX(J,K)=I_LCL$PMATRIX(J,K)-KPHTT(J,K)
THETA=FATAN2(XEST(2),XEST(1))
THETA=INT(THETA*(180./3.141592654)+.5)
RNG=SQRT(XEST(1)*XEST(1)+XEST(2)*XEST(2))
F_LCL$TMALAT=RORLAT
F_LCL$TMALON=RORLON
COSL=COS(F_LCL$TMALAT)
CALL ERB2LL(F_LCL$TMALAT,F_LCL$TMALON,RNG,THETA,0.,COSL)
IBB(KPOINT_LCL+8)=IOR(IAND(1BB(KPOINT_LCL+8),NOT(ISHFT(65535,0))),LCL00850
*ISHFT(IAND(INT(.5+(F_LCL$TMALAT--3.2)/.0001),65535),0))
IBB(KPOINT_LCL+8)=IOR(IAND(1BB(KPOINT_LCL+8),NOT(ISHFT(65535,16))))LCL00870
*,ISHFT(IAND(INT(.5+(F_LCL$TMALON--3.2)/.0001),65535),16))
CSE=FATAN2(XEST(4),XEST(3))
CSE=INT(CSE*(180./3.141592654)+.5)
SPD=SQRT(XEST(3)*XEST(3)+XEST(4)*XEST(4))

```

```

LCL00680
LCL00690
LCL00700
LCL00710
LCL00720
LCL00730
LCL00740
LCL00750
LCL00760
LCL00770
LCL00780
LCL00790
LCL00800
LCL00810
LCL00820
LCL00830
LCL00840
LCL00850
LCL00860
LCL00870
LCL00880
LCL00890
LCL00900
LCL00910

```

```

IBB(KPOINT_LCL+5)=IOR(IAND(IBB(KPOINT_LCL+5),NOT(ISHFT(511,0))),ISLCL00920
*HFT(IAND((CSE),511),0)) LCL00930
IBB(KPOINT_LCL+4)=IOR(IAND(IBB(KPOINT_LCL+4),NOT(ISHFT(4095,16))),LCL00940
*ISHFT(IAND((SPD),4095),16)) LCL00950
CONST1=I_LCL$PMATRIX(1,1)-I_LCL$PMATRIX(2,2) LCL00960
CONST2=I_LCL$PMATRIX(1,2)*I_LCL$PMATRIX(1,2) LCL00970
CONST=SQRT(CONST1*CONST1/4.+CONST2) LCL00980
I_LCL$SIGMA$QR=(I_LCL$PMATRIX(1,1)+I_LCL$PMATRIX(2,2))/2.+CONST LCL00990
I_LCL$2SIGMA=SQRT(I_LCL$SIGMA$QR)*2./2025. LCL01000
IF (I_LCL$2SIGMA.LE.500) THEN LCL01010
    I_LCL$TMAQUALITY=2 LCL01020
ELSE IF (I_LCL$2SIGMA.GT.500.AND.I_LCL$2SIGMA.LE.1000) THEN LCL01030
    I_LCL$TMAQUALITY=1 LCL01040
ELSE LCL01050
    I_LCL$TMAQUALITY=0 LCL01060
END IF LCL01070
IBB(KPOINT_LCL+9)=IOR(IAND(IBB(KPOINT_LCL+9),NOT(ISHFT(3,4))),ISHFTCL01080
*T(IAND((I_LCL$TMAQUALITY),3),4)) LCL01090
FETURN LCL01100
END LCL01110
SUBROUTINE MUL4X4(A,B,C) LCL01120
    DIMENSION A(4,4),B(4,4),C(4,4) LCL01130
    DO 60000 I=1,4 LCL01140
        DO 60001 J=1,4 LCL01150

```

```

S=0.
DO 60002 K=1,4
60002 S=S+A(I,K)*B(K,J)
60001 C(I,J)=S
60000 CONTINUE
      RETURN
      END

```

```

LCL01160
LCL01170
LCL01180
LCL01190
LCL01200
LCL01210
LCL01220

```

APPENDIX C
MULTIPLE SENSOR MODEL (RATIONAL FORTRAN)

```

Subroutine CORSNR
#####
#
# Purpose: (1) Correlate all sonar contacts (active and passive) and
#           store the updated FIX (Posit,CSE,SPD).
#
# (2) Determine a TMA quality based on the criteria:
#       If the semi-major axis of the area of probability is:
#       (a) <= 500 yds
#           TMA quality is GOOD
#       (b) > 500 yds & <= 1000 yds
#           TMA quality is FAIR
#       (c) > 1000 yds
#           TMA quality is POOR
#
# Called by: WARCYC
#
# Calls: CORR_SORT  MUL4X4  ARB2LL
#
#####

```

SNR00010
#####SNR00020
SNR00030
SNR00040
SNR00050
SNR00060
SNR00070
SNR00080
SNR00090
SNR00100
SNR00110
SNR00120
SNR00130
SNR00140
SNR00150
SNR00160
SNR00170
SNR00180
#####SNR00190

```

SNR00200
SNR00210
SNR00220
SNR00230
SNR00240
SNR00250
SNR00260
SNR00270
SNR00280
SNR00290
SNR00300
SNR00310
SNR00320
SNR00330
SNR00340
SNR00350
SNR00360
SNR00370
SNR00380
SNR00390
SNR00400
SNR00410
SNR00420

EBcommon
CORR$common

real XEST(4),X(4)
  PHI(4,4)
  PHIT(4,4)
  P(4,4),I_LCL$PMATRIX(4,4)
  H(4)
  KG(4),KGAIN(4,2)
  Z(2)
  R(2,2)

  #System Model State Vector
  #System Model Transition Matrix
  #Transpose of Transition Matrix
  #Error Covariance Matrix
  #Measurement Model Observation Matrix
  #Kalman Gain
  #Measurement Vector
  #Measurement Noise

  #Loop thru Remote Table
  for(RMT$Pointer$First; RMT$Pointer$Valid; RMT$Pointer$Next)
  {
    if(xRMT$InUse$I==$no)next      #Finf the right slots
    RMT$DetectionType$I=xRMT$DetectionType$I
    if(RMT$DetectionType$I==$Sonar$Code)      #If sonar, set composition
      putRMT$Composition$I(1)      #      to 1
  }

  if(Correlate$Sonar==$no) return

```

```

SNR 00430
#-----
SNR 00440
#Loop for each BLUE/ORANGE view
SNR 00450
SNR 00460
SNR 00470
SNR 00480
SNR 00490
SNR 00500
SNR 00510
SNR 00520
SNR 00530
SNR 00540
SNR 00550
SNR 00560
SNR 00570
SNR 00580
SNR 00590
SNR 00600
SNR 00610
SNR 00620
SNR 00630
SNR 00640
SNR 00650
SNR 00660

for (iview=firstBLUE$view; iview<=lastORANGE$view; iview=iview+1)
{
    VUE$Pointer$To iview      #Get to the right view

    RMT$Pointer$To xVUE$FirstRmtIdx$i      #Set first and last
    istart=RMT$Pointer      # RMT Index as
    RMT$Pointer$To xVUE$LastRmtIdx$i      # limits for loop
    iend=RMT$Pointer

    kore=0      #Initialize counter
    #Loop for each RMT slot in view
    for (RMT$Pointer=istart; RMT$Pointer<=iend; RMT$Pointer$next)
    {
        if (XRMT$InUse$i==$no) next      #Skip if not in use
        RMT$DetectionType$i=xRMT$DetectionType$i      #Get Detection Type

        if (Correlate$Sonar==$yes & RMT$DetectionType$i==$Sonar$Code)
        *continue
        else next
    }
}

```

```

COR01180
COR01190
COR01200
COR01210
COR01220
COR01230
COR01240
COR01250
COR01260
COR01270
COR01280
COR01290
COR01300
COR01310
COR01320
COR01330
COR01340
COR01350
COR01360
COR01370
COR01380
COR01390

F_POS2$LAT=FBB (KPOINT_RMT)
F_POS2$LON=FBB (KPOINT_RMT+1)
F_POS2$COSLAT=FBB (KPOINT_RMT+13)
X=F_POS2$LAT-F_POS1$LAT
Y=F_POS2$LON-F_POS1$LON
ANGPI(Y)
COSL=COS (F_POS2$LAT)
Y=.5*(F_POS2$COSLAT+F_POS1$COSLAT)*Y
THETAK=FACTAN2 (Y,X)
THETAK=INT (THETAK*(180./3.141592654)+.5)
DK=SQRT (X*X+Y*Y)
IBEAR (J1)=DK*SIN (IBEAR (J1)-THETAK)
70008 IF (.NOT. (I_RMT$DETECTIONTYPE.EQ.2)) GOTO 70018
H(1)=-SIN (THETA)/RNG
H(2)=COS (THETA)/RNG
H(3)=0.
H(4)=0.
DO 70019 JJ=1,4
PHT(JJ)=0.
DO 70019 KK=1,4
70019 PHT(JJ)=PHT(JJ)+I_RMT$PMATRIX (JJ, KK) *H (KK)
HPHT=0.

```

```

COR00940
COR00950
COR00960
COR00970
COR00980
COR00990
COR01000
COR01010
COR01020
COR01030
COR01040
COR01050
COR01060
COR01070
COR01080
COR01090
COR01100
COR01110
COR01120
COR01130
COR01140
COR01150
COR01160
COR01170

      GOTO 70011
70012 PHI(JJ, KK) = 0.
70011 CCNTINUE
      F_RMT$DELTIME = (IBB(103) - IIAST(K1)) / 60.
      PHI(1, 3) = F_RMT$DELTIME
      PHI(2, 4) = F_RMT$DELTIME
      DO 70013 JJ = 1, 4
      X(JJ) = 0.
      DO 70013 KK = 1, 4
70013 X(J) = X(J) + PHI(JJ, KK) * XEST(KK)
      DO 70014 JJ = 1, 4
      DO 70014 KK = 1, 4
70014 PHIT(JJ, KK) = PHT(KK, JJ)
      CALL MUL4X4(PHI, I_RMT$PMATRIX, P)
      CALL MUL4X4(P, PHIT, I_RMT$EMATRIX)
70015 IF(.NOT.(J.LE.KORE)) GOTO 70016
      J1 = IPNT(J)
      IF(IDTEE(K1).NE.IDTEE(J1)) GOTO 70017
      KPOINT_RMT = IRMTP(J1)
      I_RMT$DETECTIONTYPE = (IAND(ISHFT(1BB(KPOINT_RMT+8), -29), 3))
      THETA = FATAN2(X(2), X(1))
      THETA = INT(THETA * (180. / 3.141592654) + .5)
      RNG = SQRT(X(1) * X(1) + X(2) * X(2))
      IF(K.EQ.J) GOTO 70008

```

```

COR00700
COR00710
COR00720
COR00730
COR00740
COR00750
COR00760
COR00770
COR00780
COR00790
COR00800
COR00810
COR00820
COR00830
COR00840
COR00850
COR00860
COR00870
COR00880
COR00890
COR00900
COR00910
COR00920
COR00930

J=K
70009 IF (.NOT. (K.LT.KORE)) GOTO 70010
K1=IPNT(K)
KPOINT_RMT=IRMT(K1)
F_RMT$TMALAT=(IAND(ISHFT( IBB(KPOINT_RMT+8) ,-0) ,65535) *1.*.0001-3.2COR00740
*)
F_RMT$TMALON=(IAND(ISHFT( IBB(KPOINT_RMT+8) ,-16) ,65535) *1.*.0001-3.COR00760
*2)
F_RMT$TMACSE=(IAND(ISHFT( IBB(KPOINT_RMT+5) ,-0) ,511) )
F_RMT$TMASPD=(IAND(ISHFT( IBB(KPOINT_RMT+4) ,-16) ,4095) )
F_POS1$LAT=FBB(KPOINT_RMT)
F_POS1$LON=FBB(KPOINT_RMT+1)
F_POS1$COSLAT=FBB(KPOINT_RMT+13)
XEST(1)=F_RMT$TMALAT-F_POS1$LAT
DELLON=F_RMT$TMALON-F_POS1$LON
ANGPI(DELLON)
COSL=COS(F_RMT$TMALAT)
XEST(2)=.5*(COSL+F_POS1$CCSLAT)*DELLON
XEST(3)=F_RMT$TMASPD*COS(F_RMT$TMACSE)
XEST(4)=F_RMT$TMASPD*SIN(F_RMT$TMACSE)
DO 70011 JJ=1,4
DO 70011 KK=1,4
IF (.NOT. (JJ.EQ.KK)) GOTO 70012
PHI(JJ, KK) =1.

```

```

COR00460
COR00470
COR00480
COR00490
COR00500
COR00510
COR00520
COR00530
COR00540
COR00550
COR00560
COR00570
COR00580
COR00590
COR00600
COR00610
COR00620
COR00630
COR00640
COR00650
COR00660
COR00670
COR00680
COR00690

*83))
IEND=KPOINT_RMT
KORE=0
KPOINT_RMT=ISTART
70004 IF (.NOT. (KPOINT_RMT.LE.IEND)) GOTO 70005
IF (IBB (KPOINT_RMT+8) .EQ. 0) GOTO 70006
I_RMT$DETECTIONTYPE= (IAND (ISHFT (IBB (KPOINT_RMT+8) ,-29) , 3) )
IF (IBB (256) .EQ. 1.AND. (I_RMT$DETECTIONTYPE.EQ. 2.OR. I_RMT$DETECTIONTYPE.EQ. 3))
*YPE.EQ. 24)) GOTO 70007
GOTO 70006
70007 IF (KORE.GE. 800) GOTO 70005
KORE=KORE+1
IRMT (KORE)=KPOINT_RMT
IDTOR (KORE)= (IAND (ISHFT (IBB (KPOINT_RMT+8) ,-10) , 1023) )
IDTEE (KORE)= (IAND (ISHFT (IBB (KPOINT_RMT+7) ,-0) , 1023) )
ILAST (KORE)= (IAND (ISHFT (IBB (KPOINT_RMT+2) ,-0) , 65535) )
IBEAR (KORE)= (IAND (ISHFT (IBB (KPOINT_RMT+7) ,-10) , 511) )
IRNGE (KORE)= (IAND (ISHFT (IBB (KPOINT_RMT+7) ,-0) , 511) )
IPNT (KORE)=KORE
70006 KPOINT_RMT=KPOINT_RMT+15
GOTO 70004
70005 IF (KORE.EQ. 0) GOTO 70099
CALL CORR_SORT
K=1

```

```

COR00220
IF (IBB(KPOINT_RMT+8).EQ.0) GOTO 70002
IBB(KPOINT_RMT+10)=IOR(IAND (IBB(KPOINT_RMT+10),NOT (ISHFT(1,29))),ICOR00230
COR00240
*SHFT(IAND((0),1),29))
IBB(KPOINT_RMT+10)=IOR(IAND (IBB(KPOINT_RMT+10),NOT (ISHFT(1,28))),ICOR00250
COR00260
*SHFT(IAND (('00000001'X),1),28))
IBB(111)=1
COR00270
I_RMT$DETECTIONTYPE=(IAND (ISHFT (IBB (KPOINT_RMT+8),-29),3))
COR00280
IF (.NOT. (I_RMT$DETECTIONTYPE.EQ.2.OR.I_RMT$DETECTIONTYPE.EQ.24)) GOCOR00290
COR00300
*TO 70002
IBB(KPOINT_RMT+10)=IOR(IAND (IBB(KPOINT_RMT+10),NOT (ISHFT(3,23))),ICOR00310
COR00320
*SHFT(IAND((1),3),23))
IBB(KPOINT_RMT+10)=IOR(IAND (IBB(KPOINT_RMT+10),NOT (ISHFT(1,28))),ICOR00330
COR00340
*SHFT(IAND (('00000001'X),1),28))
IBB(111)=1
COR00350
70002 KPOINT_RMT=KPOINT_RMT+15
COR00360
GOTO 70000
COR00370
70001 IF (IBB(256).EQ.0) GOTO 70099
COR00380
IVIEW=IBB(129)
COR00390
70003 IF (.NOT. (IVIEW.LE.IBB(132))) GOTO 70099
COR00400
KPCINT_VUE=IBBP(1,06)-1540+1540*IVIEW
COR00410
KPOINT_RMT=IBBP(1,56)-15+15*(IAND (ISHFT (IBB (KPOINT_VUE+1),-0),1638COR00420
COR00430
*3))
COR00440
ISTART=KPOINT_RMT
COR00450
KPOINT_RMT=IBBP(1,56)-15+15*(IAND (ISHFT (IBB (KPOINT_VUE+1),-14),1638COR00450

```

APPENDIX D
MULTIPLE SENSOR MODEL (SOURCE CODE)

```

SUBROUTINE CORSNR                                COR00010
IMPLICIT REAL*8 (A,C)                            COR00020
INTEGER IBB(1025)                                COR00030
INTEGER*2 IBBW(2,1025),IDTOR(800),IDTEE(800),ILAST(800),IBEAR(800) COR00040
INTEGER*2 IRNGE(800),IPNT(800),KORE             COR00050
INTEGER*4 IRMTP(800)                             COR00060
BYTE IBBB(4,1025)                                COR00070
REAL*8 CBB(512)                                   COR00080
REAL FBB(1025),PBB(4,1025),H(4),PHT(4),KG(4),HPHT,HPHTR,HX,PROD(4) COR00090
REAL KGAIN(4,2),I_RMT$PMATRIX(4,4),KPHTT(4,4),KKPHTT(4,4),R(2,2)  COR00100
REAL HH(2,4),HHT(4,2),PPHT(4,2),HHPHT(2,2),SUM(2,2),ADJ(2,2)     COR00110
REAL INVSUM(2,2),PHI(4,4),PHIT(4,4),P(4,4),HHX(2),PPHTT(2,4),X(4) COR00120
REAL XEST(4),Z(2),KZHX(4)                         COR00130
EQUIVALENCE (IBB,FBB,CBB,IBBW,IBBE,PBB)           COR00140
EQUIVALENCE (IBB(513),IBBE)                       COR00150
COMMON/BOARD/IEB                                  COR00160
COMMON/SCRPAD/IRMTP,IDTOR,IDTEE,ILAST,IBEAR,IRNGE,IPNT,KORE       COR00170
DATA E/.25,0.,0.,.25/                             COR00180
KPCINT_RMT=IBBP(1,56)                             COR00190
70000 IF(.NOT.((KPCINT_RMT-GE.IEBP(1,56)).AND.KPCINT_RMT.LT.(IBBP(1,56)+ICOR00200
*BBF(2,56)))))) GOTO 70001                          COR00210

```

SNR02320

SNR02330

SNR02340

return

end

#End CORSNR

```

SNR02100
SNR02110
SNR02120
SNR02130
SNR02140
SNR02150
SNR02160
SNR02170
SNR02180
SNR02190
SNR02200
SNR02210
SNR02220
SNR02230
SNR02240
SNR02250
SNR02260
SNR02270
SNR02280
SNR02290
SNR02300
SNR02310

putRMT$TMALON$f

#New estimated course
#New estimated speed

putRMT$TMACSE$f      #New FIX course
putRMT$TMASPD$f      #      and speed

#Determine semi-major axis of area of probability
#SIGMA$squared=(P11+P22)/2+SQRT{((P11-P22)+(P11-P22))/4+P12*P12}
#2SIGMA=2*SIGMA/2025 yds

if(2SIGMA <= 500 yds) then
  TMA$Quality=2      #GOOD
else if(2SIGMA > 500 yds & <= 1000 yds) then
  TMA$Quality=1      #FAIR
else(2SIGMA > 1000 yds)
  TMA$Quality=0      #POOR

} k=j

#Set pointer to next detectee
}

```

```

#Z(2)=measured range
#H*X$hat
#Z-H*X$hat      Measurement Residual
#K*(Z-H*X$hat)
#X$hat=X$hat+K*(Z-H*X$hat)  State Update

#(P*H$transpose)$transpose
#K*(P*H$transpose)$transpose
#P=P-K*(P*H$transpose)$transpose  Error Covariance Update
}

#New estimated bearing
#New estimated range

#Compute LAT/LON of BRG/RNG from ORIGIN
CALL RRB2LL (
    F_RMT$TMLAT      #ORIGIN LAT->FIX LAT (input/output)
    F_RMT$TMLON      #ORIGIN LON->FIX LON (input/output)
    RNG              #Range from ORIGIN to TARGET
    THETA            #Bearing from ORIGIN to TARGET
    0.0              #Pass zero
    COSL             #Cosine of latitude (input/output)

    putRMT$TMLAT$f   #New FIX position

```

SNR01860
 SNR01870
 SNR01880
 SNR01890
 SNR01900
 SNR01910
 SNR01920
 SNR01930
 SNR01940
 SNR01950
 SNR01960
 SNR01970
 SNR01980
 SNR01990
 SNR02000
 SNR02010
 SNR02020
 SNR02030
 SNR02040
 SNR02050
 SNR02060
 SNR02070
 SNR02080
 SNR02090

```

SNR01620
#K*(P*H$transpose)$transpose
SNR01630
#P=P-K*(P*H$transpose)$transpose      Error Covariance Update
SNR01640
SNR01650
SNR01660
SNR01670
SNR01680
SNR01690
SNR01700
#Measurement Observation Matrix
SNR01710
SNR01720
SNR01730
SNR01740
SNR01750
SNR01760
SNR01770
SNR01780
SNR01790
SNR01800
#H*P*H$transpose+R      R=BRG measurement error=1.5 degrees
SNR01810
#      # =RNG measurement error=1.5 naut. mi.
SNR01820
SNR01830
SNR01840
SNR01850
#Z(1)=measured bearing

```

```

#K*(P*H$transpose)$transpose

```

```

#P=P-K*(P*H$transpose)$transpose

```

```

Error Covariance Update

```

```

if(RMT$DetectionType != $Active$Sonar)break

```

```

#irnge(j1)=observed range adjusted to ORIGIN platform

```

```

#Measurement Observation Matrix

```

```

HH(2,1)=COS(THETA$hat)

```

```

HH(2,2)=SIN(THETA$hat)

```

```

HH(1,1)=-SIN(THETA$hat)/RNG$hat

```

```

HH(1,2)=COS(THETA$hat)/RNG$hat

```

```

HH(1,3)=HH(1,4)=HH(2,3)=HH(2,4)=0

```

```

#H$transpose

```

```

#P*H$transpose

```

```

#H*P*H$transpose

```

```

#H*P*H$transpose+R

```

```

R=BRG measurement error=1.5 degrees

```

```

# =RNG measurement error=1.5 naut. mi.

```

```

#(H*P*H$transpose+R)$inverse

```

```

#Kalman Gain

```

```

#Z(1)=measured bearing

```

```

#THETAK=bearing from ORIGIN to DETECTOR
#DK=distance from ORIGIN to DETECTOR

ibear(j1)=DK*SIN(OBS BRG - BRG from ORIGIN) #Observed bearing SNR01380
#adjusted to SNR01390
#ORIGIN platform SNR01400
SNR01410
SNR01420
SNR01430
SNR01440
SNR01450
if(RMT$DetectionType != $Passive$Sonar) check for $Active$SonarSNR01460
SNR01470
#Measurement Observation Matrix
H(1)=-SIN(THETA$hat)/RNG$hat #H=H$transpose SNR01480
H(2)=COS(THETA$hat)/RNG$hat SNR01490
H(3)=H(4)=0 SNR01500
SNR01510
#P*H$transpose = (P*H$transpose)$transpose SNR01520
#H*P*H$transpose SNR01530
#H*P*H$transpose+R R=BRG measurement error=1.5 degrees SNR01540
#Kalman Gain SNR01550
SNR01560
#Z(1)=measured bearing SNR01570
#H*X$hat SNR01580
#ZX=Z(bearing)-H*X$hat Measurement Residual SNR01590
#X*(Z-H*X$hat) SNR01600
#X$hat=X$hat+K*(Z-H*X$hat) State Update SNR01610

```

```

#MEASUREMENT:
    for(j=k; j<=kore; j=j+1)
    {
        j1=ipnt(j)

        if(idtee(k1) != idtee(j1)) break

        KPOINT_RMT=irmtp(j1)      #Set pointer

        RMT$DetectionType=xRMT$DetectionType      #Get Detection Type

        #Estimated bearing
        #Estimated range

        if(k=j)
            #ORIGIN platform
            POS2$LAT$F=xPOS2$LAT$F      #Get posit of next detector
            POS2$LON$F=xPOS2$LON$F
            POS2$COSLAT$F=xPOS2$COSLAT$F

        #Adjust contact bearing to ORIGIN
        #X=north-south distance from ORIGIN to DETECTOR
        #Y=east-west distance from ORIGIN to DETECTOR

```

```

RMT$TMACSE$F=XRMT$TMACSE$F      SNR00910
RMT$TMASPD$F=XRMT$TMASPD$F      SNR00920
                                  SNR00930
                                  SNR00940
                                  SNR00950
                                  SNR00960
                                  SNR00970
                                  SNR00980
                                  SNR00990
                                  SNR01000
                                  SNR01010
                                  SNR01020
                                  SNR01030
                                  SNR01040
                                  SNR01050
                                  SNR01060
                                  SNR01070
                                  SNR01080
                                  SNR01090
                                  SNR01100
                                  SNR01110
                                  SNR01120
                                  SNR01130

      POS1$LAT$F=XPOS1$LAT$F      #Get posit of Detector
      POS1$LON$F=XPOS1$LON$F
      POS1$COSLAT$F=XPOS1$COSLAT$F

#MOVEMENT:

      #State Extrapolation
      XEST(1)=X      #Distance in north-south direction
      ANGPI(DELLON)  #Insure shortway around earth
      XEST(2)=Y      #Distance in east-west direction
      XEST(3)=X$dot  #Speed vector in north-south direction
      XEST(4)=Y$dot  #Speed vector in east-west direction

      #Initialize the PHI matrix
      PHI(1,3)=PHI(2,4)=$delta$t  #Movement time in hrs since
                                   # last update

      X$hat=PHI*X$hat      State Extrapolation Vector

      PHI$transpose
      P=PHI*p*PHI$transpose  Error Covariance Extrapolation

```


DO 70020 JJ=1,4	COR01400
70020 HPHT=HPHT+H(JJ)*PHT(JJ)	COR01410
HPHTR=HPHT+.25	COR01420
DO 70021 JJ=1,4	COR01430
70021 KG(JJ)=PHT(JJ)/HPHTR	COR01440
Z(1)=FLOAT(IBEAR(J1))	COR01450
HX=0.	COR01460
DO 70022 JJ=1,4	COR01470
70022 HX=HX+H(JJ)*X(JJ)	COR01480
ZHX=Z(1)-HX	COR01490
DO 70023 JJ=1,4	COR01500
70023 PROD(JJ)=KG(JJ)*ZHX	COR01510
DO 70024 JJ=1,4	COR01520
70024 X(JJ)=X(JJ)+PROD(JJ)	COR01530
DO 70025 JJ=1,4	COR01540
DO 70025 KK=1,4	COR01550
70025 KPHTT(JJ, KK)=KG(JJ)*PHT(KK)	COR01560
DO 70026 JJ=1,4	COR01570
DO 70026 KK=1,4	COR01580
70026 I_RMT\$PMATRIX(JJ, KK)=I_RMT\$PMATRIX(JJ, KK)-KPHTT(JJ, KK)	COR01590
GOTO 70027	COR01600
70018 IF(-NOT.(I_RMT\$DETECTIONTYPE.EQ.24))GOTO 70027	COR01610
IF(K.EQ.J)GOTO 70028	COR01620

```

    IRNGE(J1)=SQRT(DK*DK+IRNGE(J1)*IRNGE(J1))
70028 HH(2,1)=COS(THETA)
    HH(2,2)=SIN(THETA)
    HH(1,1)=-H(2,2)/RNG
    HH(1,2)=H(2,1)/RNG
    HH(1,3)=0.
    HH(1,4)=0.
    HH(2,3)=0.
    HH(2,4)=0.
    DO 70029 JJ=1,4
    DO 70029 KK=1,4
70029 HHT(JJ, KK)=HH(KK, JJ)
    DO 70030 JJ=1,4
    DO 70031 KK=1,2
    S=0.
    DO 70032 LL=1,4
70032 S=S+I_RMT$PMATRIX(JJ, LL)*HHT(LL, KK)
70031 PPHT(JJ, KK)=S
70030 CONTINUE
    DO 70033 JJ=1,2
    DO 70034 KK=1,2
    S=0.
COR01630
COR01640
COR01650
COR01660
COR01670
COR01680
COR01690
COR01700
COR01710
COR01720
COR01730
COR01740
COR01750
COR01760
COR01770
COR01780
COR01790
COR01800
COR01810
COR01820
COR01830
COR01840

```

COR01850
COR01860
COR01870
COR01880
COR01890
COR01900
COR01910
COR01920
COR01930
COR01940
COR01950
COR01960
COR01970
COR01980
COR01990
COR02000
COR02010
COR02020
COR02030
COR02040
COR02050
COR02060
COR02070
COR02080

DO 70035 LL=1,4
70035 S=S+HH(JJ,LL)*PPHT(LL,KK)
70034 HHPHT(JJ,KK)=S
70033 CONTINUE
DO 70036 JJ=1,2
DO 70036 KK=1,2
70036 SUM(JJ,KK)=HHPHT(JJ,KK)+R(JJ,KK)
DET=SUM(1,1)*SUM(2,2)-SUM(1,2)*SUM(2,1)
ADJ(1,1)=SUM(2,2)
ADJ(1,2)=-SUM(1,2)
ADJ(2,1)=-SUM(2,1)
ADJ(2,2)=SUM(1,1)
DO 70037 JJ=1,2
DO 70037 KK=1,2
70037 INVSUM(JJ,KK)=ADJ(JJ,KK)/DET
DO 70038 JJ=1,4
DO 70039 KK=1,2
S=0.
DO 70040 LL=1,2
70040 S=S+PPHT(JJ,LL)*INVSUM(LL,KK)
70039 KGAIN(JJ,KK)=S
70038 CONTINUE
Z(1)=FLOAT(IBEAR(J1))
Z(2)=FLOAT(IRNGE(J1))

COR02090
COR02100
COR02110
COR02120
COR02130
COR02140
COR02150
COR02160
COR02170
COR02180
COR02190
COR02200
COR02210
COR02220
COR02230
COR02240
COR02250
COR02260
COR02270
COR02280
COR02290
COR02300
COR02310
COR02320

DO 70041 JJ=1,2
S=0.
DO 70042 KK=1,4
70042 S=S+HH(JJ, KK)*X(KK)
70041 HHX(JJ)=S
DO 70043 JJ=1,2
70043 HHX(JJ)=Z(JJ)-HHX(JJ)
DO 70044 JJ=1,4
S=0.
DO 70045 KK=1,2
70045 S=S+KGAIN(JJ, KK)*HHX(KK)
70044 KZHX(JJ)=S
DO 70046 JJ=1,4
70046 X(JJ)=X(JJ)+KZHX(JJ)
DO 70047 JJ=1,2
DO 70047 KK=1,4
70047 PPHTT(JJ, KK)=PPHT(KK, JJ)
DO 70048 JJ=1,4
DO 70049 KK=1,4
S=0.
DO 70050 LL=1,2
70050 S=S+KGAIN(JJ, LL)*PPHTT(LL, KK)
70049 KKPHTT(JJ, KK)=S
70048 CCNTINUE

```

DO 70051 JJ=1,4
DO 70051 KK=1,4
70051 I_RMT$PMATRIX(JJ, KK)=I_RMT$PMATRIX(JJ, KK)-KKPHTT(JJ, KK)
70027 J=J+1
GOTO 70015
70017 THETA=FATAN2(X(2), X(1))
THETA=INT(THETA*(180./3.141592654)+.5)
RNG=SQRT(X(1)*X(1)+X(2)*X(2))
F_RMT$THALAT=F_POS1$LAT
F_RMT$THALON=F_POS1$LON
COSL=F_POS1$COSIAT
CALL RRB2LL(F_RMT$THALAT, F_RMT$THALON, RNG, THETA, 0., COSL)
KPCINT_RMT=IRMT(K1)
IBB(KPOINT_RMT+8)=IOR(IAND(1BB(KPOINT_RMT+8), NOT(ISHFT(65535, 0))), COR02460
*ISHFT(IAND(INT(.5+(F_RMT$THALAT--3.2)/.0001), 65535), 0))
IBB(KPOINT_RMT+8)=IOR(IAND(1BB(KPOINT_RMT+8), NOT(ISHFT(65535, 16))), COR02480
*, ISHFT(IAND(INT(.5+(F_RMT$THALON--3.2)/.0001), 65535), 16))
CSE=FATAN2(X(4), X(3))
CSE=INT(CSE*(180./3.141592654)+.5)
SPD=SQRT(X(3)*X(3)+X(4)*X(4))
IBB(KPOINT_RMT+5)=IOR(IAND(1BB(KPOINT_RMT+5), NOT(ISHFT(511, 0))), ISCOR02530
*HFT(IAND((CSE), 511), 0))
IBB(KPOINT_RMT+4)=IOR(IAND(1BB(KPOINT_RMT+4), NOT(ISHFT(4095, 16))), COR02550
*ISHFT(IAND((SPD), 4095), 16))
COR02330
COR02340
COR02350
COR02360
COR02370
COR02380
COR02390
COR02400
COR02410
COR02420
COR02430
COR02440
COR02450
COR02460
COR02470
COR02480
COR02490
COR02500
COR02510
COR02520
COR02530
COR02540
COR02550
COR02560

```

```

COR02570
CONST1=I_RMT$PMATRIX(1,1)-I_RMT$PMATRIX(2,2)
COR02580
CCNST2=I_RMT$PMATRIX(1,2)*I_RMT$PMATRIX(1,2)
COR02590
CONST=SQRT(CONST1*CONST1/4.+CONST2)
COR02600
I_RMT$SIGMA$QR=(I_RMT$PMATRIX(1,1)+I_RMT$PMATRIX(2,2))/2.+CONST
COR02610
I_RMT$2SIGMA=SQRT(I_RMT$SIGMA$QR)*2./2025.
COR02620
IF(I_RMT$2SIGMA.LE.500) THEN
COR02630
    I_RMT$TMAQUALITY=2
COR02640
ELSE IF(I_RMT$2SIGMA.GT.500.AND.I_RMT$2SIGMA.LE.1000) THEN
COR02650
    I_RMT$TMAQUALITY=1
COR02660
ELSE
COR02670
    I_RMT$TMAQUALITY=0
COR02680
END IF
COR02690
IBB(KPOINT_RMT+9)=IOR(IAND(IEB(KPOINT_RMT+9),NOT(ISHFT(3,4))),ISHFCOR02690
COR02700
    *T(IAND((I_RMT$TMAQUALITY),3),4))
COR02710
70016 K=J
COR02720
    GOTO 70009
COR02730
70010 IVIEW=IVIEW+1
COR02740
    GOTO 70003
COR02750
70099 RETURN
COR02760
    END

```

BIBLIOGRAPHY

B-K Dynamics, Inc., Battle Group Training Computer Support Facility (BGTCSF) Passive Schar T&E Plan and Results, prepared for Naval Ocean Systems Center, San Diego, California, September 1983.

B-K Dynamics, Inc., Battle Group Training Computer Support Facility (BGTCSF) Target Motion Analysis, prepared for Naval Ocean Systems Center, San Diego, California, June 1983.

Pacer Systems, Inc., Interim Battle Group Tactical Trainer (IBGTT) Instructor User's Guide, Volume I, prepared for Naval Ocean Systems Center, San Diego, California, 17 January 1983.

Shudde, Rex H., A Multiple Leg TMA Procedure with Programs for the Hewlett-Packard HP-41CV, the Hewlett-Packard HP-75C, the Sharp PC-1500 (TRS-80 PC-2), and the Radio Shack TRS-80 Model 100 Portable Computers, Naval Postgraduate School, Monterey, California, September 1983.

System Development Corp., NWISS Programmer/Analyst Guide, prepared for Naval Ocean Systems Center, San Diego, California, 31 May 1984.

INITIAL DISTRIBUTION LIST

	No.	Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22314		2
2. Library, Code 0142 Naval Postgraduate School Monterey, CA 93943		2
3. Commander Naval Ocean Systems Center ATTN: Code 41 San Diego, CA 92152		1
4. LT. Keith N. Spangenberg Cruiser-Destroyer Group Five FPO, San Francisco 96601-4703		1

END

FILMED

8-85

DTIC