U.S. Coast Guard Research and Development Center
1082 Shennecossett Road, Groton, CT 06340-6096

Report No. CG-D-16-99

Maritime Operations Simulation Model: Search and Rescue (SAR) Application Report

FINAL REPORT
MAY 1999

Prepared for:

U.S. Department of Transportation
United States Coast Guard
Operations (G-O)
Washington, DC 20593-0001

N O T I C E

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report.

This report does not constitute a standard, specification, or regulation.

Marc B. Mandler, Ph.D.
Technical Director
U.S. Coast Guard
Research & Development Center
1082 Shennecossett Road
Groton, CT 06340-6096

Technical Report Documentation Page

1. Report No.	2. Government Accession Number	3. Recipient's Catalog No.
CG-D-16-99		

4. Title and Subtitle	5. Report Date
Maritime Operations Simulation Model: Search and Rescue (SAR) Application Report	May 1999
	6. Performing Organization Code: Project 1012.3.3

7. Author(s)	8. Performing Organization Report No.
Kevin Downer	R&DC-122-99

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
U.S. Coast Guard Research and Development Center 1082 Shennecossett Road Groton, CT 06340-6096	N/A
	11. Contract or Grant No.: N/A

12. Sponsoring Organization Name and Address	13. Type of Report & Period Covered
U.S. Department of Transportation United States Coast Guard Office of Search and Rescue Washington, DC 20593-0001	Final Report
	14. Sponsoring Agency Code: Commandant (G-OPR) U.S. Coast Guard Headquarters Washington, DC 20593-0001

15. Supplementary Notes

The R&D Center's technical point of contact is Mr. Kevin Downer, 860-441-254, kdowner@rdc.uscg.mil.

16. Abstract (MAXIMUM 200 WORDS)

The Search and Rescue (SAR) application is a modeling effort that was conducted in conjunction with the development of the Coast Guard's Maritime Operations Simulation (MarOpsSim) tool. The main objective of the MarOpsSim project is to create an affordable, internally sustainable, simulation modeling tool which can be applied to multiple Coast Guard mission areas. This SAR application work supported the initial development of MarOpsSim and begins to address an identified lack of analysis tools available for conducting operational effectiveness studies in SAR. Presently, no suitable modeling and analysis tools exist to study various alternatives and possibilities for improved search planning models, technology, tactics and doctrine. This work seeks to exercise MarOpsSim and provides a simulation baseline for conducting future SAR mission analysis studies. Results reported herein show that MarOpsSim can effectively be configured for the analysis of the SAR mission when detection capabilities, expressed as a lateral range curve, are assumed.

17. Key Words	18. Distribution Statement
Maritime Operations Simulation, Search and Rescue, Lateral Range Curve	This document is available to the U.S. public through the National Technical Information Service, Springfield, VA 22161

19. Security Class (This Report)	20. Security Class (This Page)	21. No of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED		

Form DOT F 1700.7 (8/72) Reproduction of form and completed page is authorized.

ACKNOWLEDGEMENT

This report is based upon material and documentation provided by MicroSystems Integration, Inc., Pawcatuck Connecticut, and Sonalysts, Inc., Waterford, Connecticut (Contract No: DTCG39-97-D-E00062). MicroSystems is the prime contractor supporting the development of the Maritime Operations Simulation.

Assistance in developing the formulations for modeling lateral range curve detection was provided by Mr. Dave Larson of Analysis and Technology, Inc., North Stonington, Connecticut. Assistance in developing validation criteria and search pattern design was provided by LCDR Christopher Rodriguez and LT Harlan Wallace.

Executive Summary

The Search and Rescue (SAR) application is a modeling effort that was conducted by the United States Coast Guard Research and Development Center (R&DC) in conjunction with the development of the Maritime Operations Simulation (MarOpsSim). MarOpsSim is a discrete event simulation designed to model the common maritime operations of the Coast Guard. MarOpsSim is intended to provide the Coast Guard with a cost-effective simulation modeling capability supporting mission analysis and acquisition efforts. This report documents an interim stage of MarOpsSim development aimed at modeling specific aspects of the Coast Guard's SAR mission.

The chief motivation for the SAR application was to address the identified lack of analysis tools available for conducting operational effectiveness studies in SAR. Presently, no suitable modeling and analysis tools exist to study and prioritize various alternatives and possibilities for improved search planning models, technology, tactics and doctrine. This work seeks to exercise MarOpsSim and provide a simulation baseline for conducting future SAR mission analysis studies.

The SAR application development was structured in two phases. The objective of phase one was to develop and analyze a scenario that models Coast Guard resources engaged in visual searching, with target detection capability characterized by lateral range curves (LRCs). LRCs are curves plotting cumulative detection probabilities (P_D) versus the closest point of approach (CPA). The second phase extended the work in phase one to accommodate sensors whose detection performance could be represented by a generalized lateral range function. In both phases the sensors being modeled were visual sensors; however, the approach used to generalize a visual sensor could be extended to non-visual sensors such as radar. This work also generated scenarios and tactics that have been incorporated into the MarOpsSim baseline and are available for further experimentation.

Results from exercising the developed scenarios are summarized below:

- The simulated search platform produced detection results consistent with a known LRC.
- A method for generalizing the LRC detection capability was successfully established. Simulation runs conducted with the LRC detection model demonstrated appropriate sensitivity to its driving parameters and produced a reasonable representation of the probability of detection versus coverage factor curve derived from search theory.
- Methods were developed in this effort that allowed for the creation of an accurate representation of the decisions and tactics involved in SAR planning and response.
- The analysis of resource and target motion provided reasonable and expected results.

MarOpsSim is still an evolving product with tremendous potential. Tasks related to the SAR application that should follow this work include:

- Independent verification and validation of the core simulation, the input and output processes, and the scenarios developed under this effort.

- Development of an approach for determining the time of detection when using lateral range curves as a detection model. This is expected to be an important factor in applications where the time of detection can impact search unit tactics and scenario results.
- The creation of a validated tactics library, which could provide the basis for using simulation to explore the effectiveness of SAR plans, tactics, capability enhancements and resource allocation schemes.

MarOpsSim is presently being extended to model law enforcement mission scenarios. In addition, the model will be undergoing further development for use in the Coast Guard's Deepwater Acquisition effort. In light of these two efforts and the work established here, MarOpsSim has the potential to provide the Coast Guard with a powerful new capability for SAR mission analysis, training, and resource planning.

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

1. INTRODUCTION

1.1 Background

The Search and Rescue (SAR) application is a modeling effort being conducted in conjunction with the Maritime Operations Simulation (MarOpsSim) project. MarOpsSim is under development by the R&DC in support of the Mission Analysis Division of Strategic and Business Planning Office of the Operations Resource Management Directorate for the Assistant Commandant of Operations, COMDT (G-ORP-2). The main objective of the MarOpsSim project is to create an affordable, internally sustainable, simulation modeling tool which can be applied to multiple Coast Guard mission areas. In order to build the foundation for the multi-mission capabilities, two initial applications were targeted for development. The first is the SAR application described here. The second application, which will be built upon many of the same simulation components created for SAR, is focused on Coast Guard law enforcement. The law enforcement application will be documented in a separate report.

The motivation for the SAR application, beyond the needs of MarOpsSim development, is the identified lack of analysis tools available for conducting operational effectiveness studies in SAR. Presently, no suitable modeling and analysis tools exist to study various alternatives and possibilities for improved search planning models, technology, tactics and doctrine. This work seeks to exercise MarOpsSim and provide a simulation baseline for conducting future SAR mission analysis studies.

1.2 MarOpsSim Overview

The MarOpsSim is designed to be a multi-mission simulation environment for the analysis of Coast Guard operations. The simulation models the core functionality of most Coast Guard maritime operations and provides the ability to select or define components as required for a particular mission application. These components can include mission specific platforms (cutters and aircraft), tactics, sensors, target types and command and control at variable levels of detail. The overall MarOpsSim design promotes future enhancement by providing a system that enables users to easily add and validate new capabilities by building upon previously created applications. The key to having this kind of flexibility is the use of scripts. A script, or a collection of scripts, can be thought of as written language that is interpreted by the simulation engine resulting in the execution of a scenario (a scenario is a particular instance of a mission application exercised in the simulation.). The scripting language enables the user to define the application scenario to the simulation without ever needing to change the simulation code.

The MarOpsSim consist of the simulation engine, input and results data bases, and a script parser depicted in **Figure 1.**

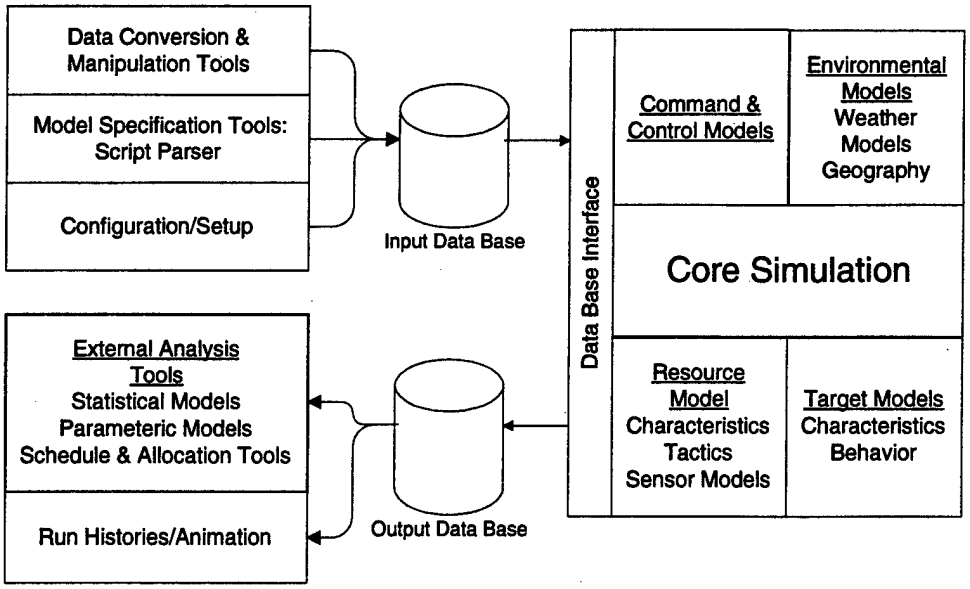

Figure 1: MarOpsSim Overview

The scripting language and the data base structure provide the means for model reuse and versatile results analysis. Its operation is simple and straightforward. The Script Parser reads the application scripts, which define the scenario for execution, and breaks the scripts into appropriate elements to populate the input database. The simulation then reads the input database and executes multiple replications of the scenario, collecting results for analysis in the output database.

1.3 SAR Application Scope and Objectives

The SAR application is structured as a two-phased effort involving the development of simulation scenarios designed to model certain aspects of SAR mission planning. The objective of phase one, the Visual Lateral Range Curve (VLRC) Search Model, was to develop and analyze a scenario that models Coast Guard resources engaged in a visual search with target detection capability characterized by lateral range curves (LRCs). LRCs are curves plotting cumulative detection probabilities (P_D) against the closest point of approach (CPA). The second phase, the Generalized Lateral Range Curve (GLRC) Search Model, extends the work on phase one to accommodate visual sensors whose detection performance is represented by a valid lateral range function. In addition, phase two explores extending MarOpsSim's scripting capabilities to produce a realistic representation of Coast Guard search operations. Both phases of this work have generated scenarios and tactics that have been incorporated into the MarOpsSim baseline and are available for further experimentation.

1.4 Development Approach

The approach for developing the SAR application involved specifying levels of achievement and then building upon them until the objectives for both phases were meet. Three scenarios with increasing levels of complexity and well-defined performance criteria were established:

1. Model a search platform with a detection capability based on a LRC. Demonstrate that a searcher/target simulation produces detections similar to the known curve.

2. Extend the model to provide a means for defining a generalized lateral range curve. Demonstrate that the simulation produces results consistent with the theoretical probability of detection versus coverage factor curve.
3. Model a realistic representation of Coast Guard search operations and tactics. Demonstrate accurate modeling of platform characteristics including motion and detection capabilities.

These scenarios set specific targets for MarOpsSim development and provide a baseline for future SAR simulation modeling and analysis. Each scenario was kept extremely simple in order to facilitate understanding of the model's results for verification and validation analysis. They are also intended to assist a novice user in learning how to work with the model.

2. VISUAL LATERAL RANGE CURVE (VLRC) SEARCH MODEL

The objective of the VLRC Search Model was to demonstrate that the MarOpsSim detection model reasonably approximates a theoretical VLRC. Theoretical curves have been developed by collecting field test data and statistically fitting a function to the data. The fitted function can be drawn as a curve representing cumulative detection probabilities versus closest point of approach ranges. Every LRC is specific to a particular combination of factors including environmental conditions, search platform attributes (sensors, speed, altitude, etc.) and target type. LRCs are often used to estimate sweep widths as an aid in search planing. An excellent discussion of LRCs and search planning can be found in The Theory of Search - A Simplified Explanation (Soza and Company, Ltd., 1996).

2.1 Model Development

A scenario was created to populate a search area with stationary targets of similar type and size. The weather and target parameters, feeding the detection algorithm, were fixed for the scenario. An aircraft was configured to fly a prescribed pattern at a fixed altitude through the field of targets. The individual detection events including time, location, and CPA were recorded for analysis. A ratio of successful detections versus potential detections was derived for each set of targets located along tracks associated with unique CPA values. These ratios are used to develop a simulated VLRC. The simulated VLRC was compared to the theoretical VLRC curve in order to validate the visual detection model.

2.1.1 Conceptual Model for Visual Lateral Range Curve Searches

The approach to modeling VLRC searches in MarOpsSim was to use an equation with externally supplied parameters corresponding to an assumed theoretical curve. This task exercised the linear logistic (Logit) family of functions.

The VLRC function is a statistical abstraction providing the P_D of a target for a particular lateral range (designated by the CPA between the target and the search platform). LRCs have been used by the Coast Guard, as a SAR planning tool, to determine search track spacing required for achieving a desired probability of detection (POD) for a given search. The VLRC can be expressed by a family of curves generated by the Logit model. The family of equations in the Logit model has the following format:

$$P_D = \frac{1}{1+e^{-\lambda}}, \qquad (1)$$

$$\text{where } \lambda = \sum_{i=1}^{n} a_i x_i .$$

Each term $a_i x_i$ in the summation is a fitting parameter that characterizes the varying conditions affecting detection, such as weather, target, and sensor characteristics. The a_1 term represents the CPA coefficient where x_1 is the CPA. A good discussion of how Logit models have been used in developing lateral range curves can be found in Factors Affecting Coast Guard SAR Unit Visual Detection Performance (Edwards, Osmer, Mazour & Hover, 1981). The specific implementation used in the analysis is described below:

$$P_D = \frac{1}{1+e^{-(-0.963\times CPA - 0.0062\times SRF + 2.546)}}, \quad (2)$$

$$\text{Where, } SRF = (WindSpeed \times WaveHeight)^{3/2}$$

In equation (2) the components of the exponential term are parameters derived from fitting the Logit function to empirical data. The term SRF is defined as a "surface roughness factor" and is used only as a convenient way to aggregate the wind speed and wave height terms.

When the simulation is run the weather and target parameters are fixed for the specified curve. Detection occurs when a uniform random number, drawn for each target/searcher pair, is less than the derived P_D value. Thus, the VLRC simulation will produce a list of detections and misses associated with the encounter between each target/searcher pair. Analysis is performed by grouping successful detections by CPA and generating the ratio of detections to total targets. The resulting ratio can be compared with the original P_D estimates for each CPA, generated from the Logit equation.

2.2 Stationary Target Scenario

Setting up the Stationary Target scenario involves:

- Scripting a functioning scenario, including identification of platforms, their attributes and behaviors (tactics),
- Running the scenario through the simulation and
- Analyzing the results produced by the simulation.

2.2.1 Scenario Specification

The scenario components modeled for the simulation runs include:

- **The geography** – approximately a 12- by 46-nautical-mile rectangular region of open water; defined from 40.00 N, 74.00 W, to 40.20 N, 73.00 W, as depicted in **Figure 2.**
- **A searching helicopter** (chosen to match empirical data) assigned to a fly-by tactic defined as a straight line (due east) flown down the center of the geographic region
- **25,000 generic stationary targets** located on a grid spaced so that targets lie evenly along lines spaced ≈ .00127 degrees latitude apart perpendicular to the track of the search unit.
- **The following events** – start-of-simulation, detection, end-of-simulation
- **Weather** – as described using the following parameters:
  1. **Wave height** – real number expressing the height at which one-third of the waves exceed (used in the Logit formulation)
  2. **Wind speed** – real number expressing the speed of the wind in knots (used in the Logit formulation)
  3. **Visibility** – real number acting as a placeholder for future models

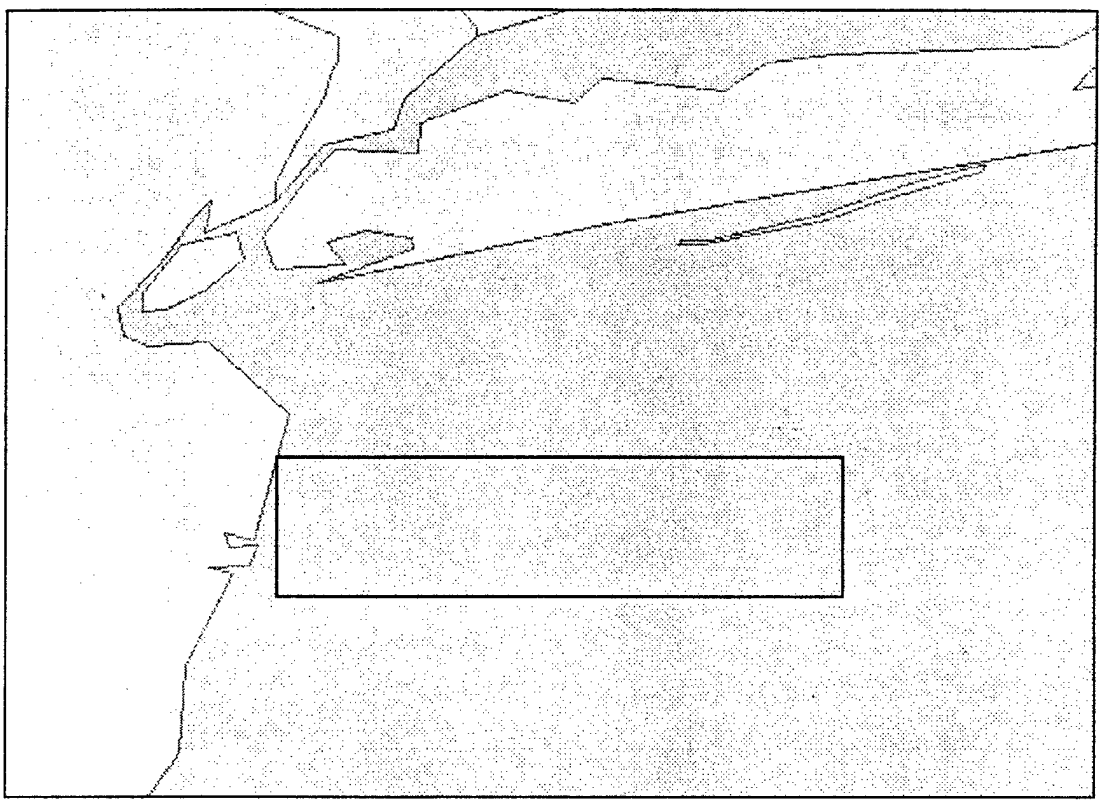

Figure 2: VLRC Setup

No resource scheduler or course changes are required, because the Helicopter flies a straight line from the beginning to the end of the simulation, making and recording detections along the way.

2.2.2 Scenario Analysis

Four iterations were conducted, producing an average of 1266 targets at 79 unique CPA ranges. Inputs included fixed weather conditions associated with the wave height and wind speed parameters for the Logit function established for this exercise (specifically, wind speed = 15 knots and wave height = 2 feet). These parameters produced inputs presented in **Table 1: VLRC Input Table.** The columns of Table 1 are divided into lateral ranges [LATRNG], cumulative probability of detection at varying CPAs derived from equation (2) [P_D], wind speed [WIND], wave height [HS], altitude (height of eye) [ALT] and the [SRF] defined in equation (2) above.

Table 1: VLRC Input Table

LATRNG	P_D	WIND	HS	ALT	SRF
0.0	0.822	15	2	500	164.31
0.5	0.740	15	2	500	164.31
1.0	0.637	15	2	500	164.31
1.5	0.521	15	2	500	164.31
2.0	0.402	15	2	500	164.31
2.5	0.293	15	2	500	164.31
3.0	0.204	15	2	500	164.31
3.5	0.137	15	2	500	164.31
4.0	0.089	15	2	500	164.31
4.5	0.057	15	2	500	164.31
5.0	0.036	15	2	500	164.31
5.5	0.023	15	2	500	164.31
6.0	0.014	15	2	500	164.31
6.5	0.009	15	2	500	164.31
7.0	0.005	15	2	500	164.31
7.5	0.003	15	2	500	164.31

The theoretical LRC is presented in **Figure 3,** where the P_D is plotted along the vertical axis and the CPA along the horizontal axis.

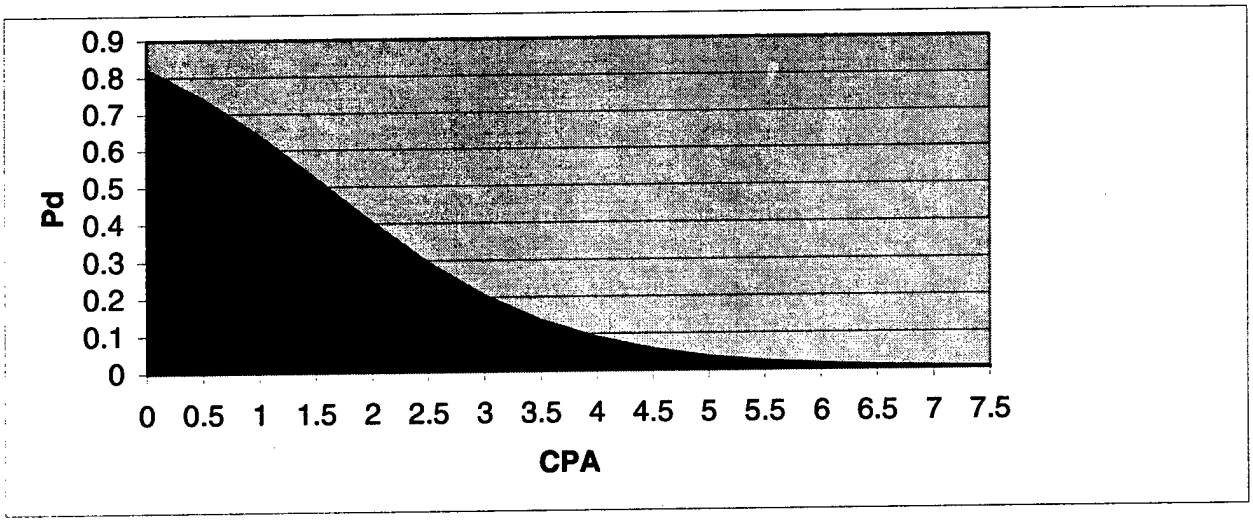

Figure 3: Theoretical VLRC

Simulation output includes the target identification number, location, and CPA, along with the time of each detection. Subsequent analysis includes generating a VLRC from simulation results and plotting the output data in a geographical information system (GIS). The VLRC is generated by forming detection probabilities (number of detections/total targets) for the vessels at each CPA and plotting the results as a curve. The GIS display consists of a plot of the location of each target overlaid with a plot of the detections from the first iteration of the simulation.

2.3 Results

Of the 100,000 targets generated over the four iterations, 29,350 were detected. The output associated with these detections was analyzed as described above. The resulting LRC and GIS plots are presented in this section, along with some generalizations drawn from these results.

Simulation results included a listing of detections that were parsed into CPA ranges. The ratio of number of detections to total targets for each CPA produced a table of P_Ds, which can be found in **Appendix A: VLRC Validation Runs.** The data from that table was then plotted in Microsoft Excel ©. The resulting curve is illustrated in **Figure 4.**

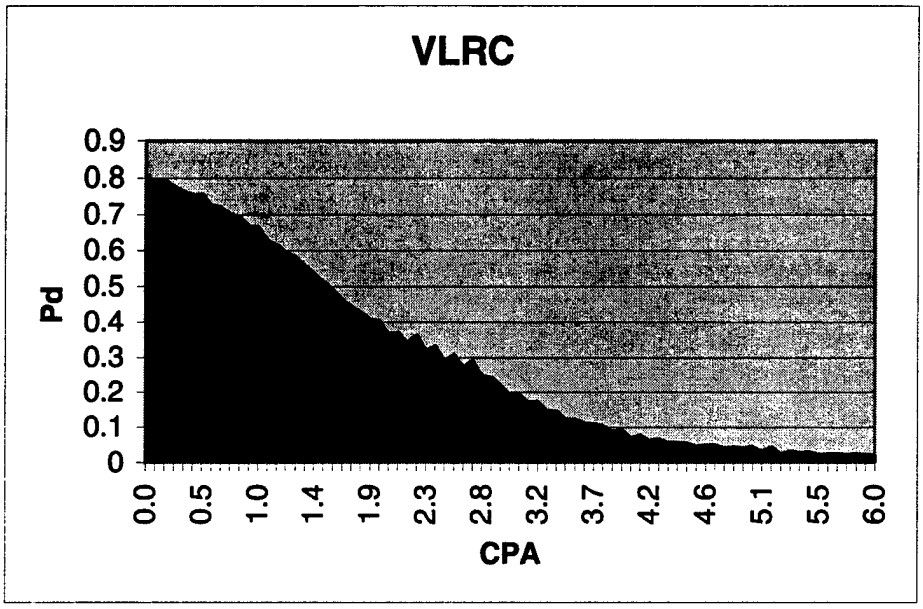

Figure 4: Simulated VLRC Results

Figure 5 compares the simulation results to those expected from the theoretical model.

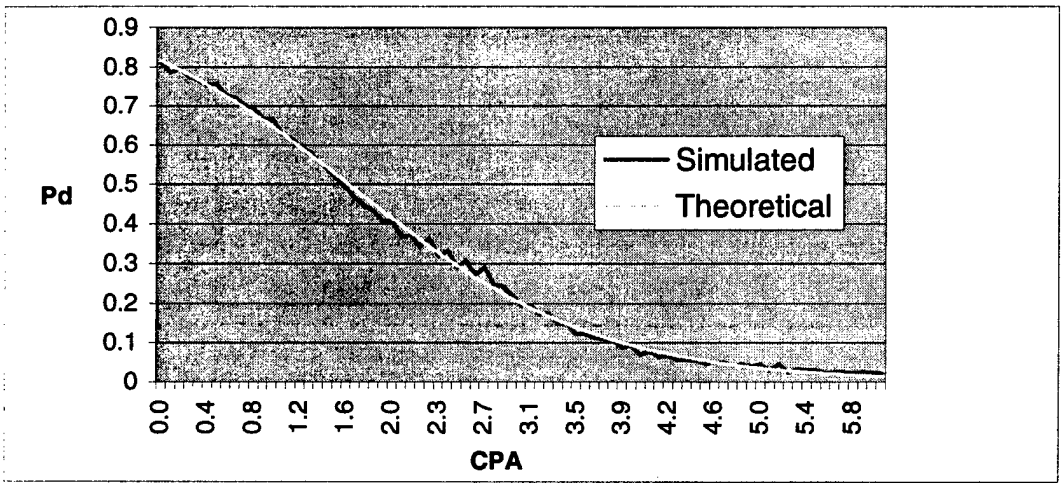

Figure 5: Comparative VLRC Curves

Initial target locations and detection locations from the first run were plotted in a GIS. **Figure 6** illustrates the results.

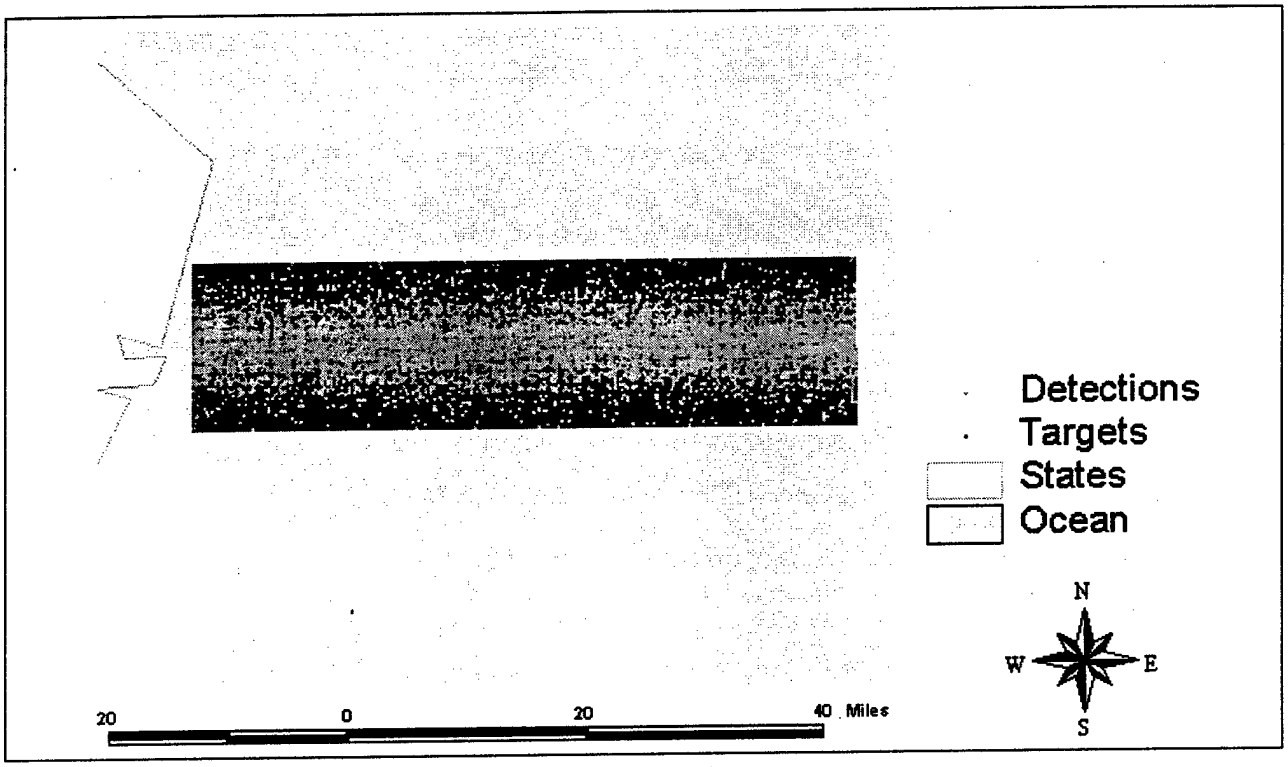

Figure 6: Simulated Detections Plot

Analysis of results support the following conclusions:

- Both the GIS and VLRC graphs illustrate that a specific form of VLRC can be effectively modeled by MarOpsSim.
- MarOpsSim successfully implemented a simple target search scenario demonstrating appropriate kinematics and target searcher interactions.

3. GENERALIZED LATERAL RANGE CURVE (GLRC) SEARCH MODEL

The objective of the GLRC Model was to extend the work conducted under the VLRC task by developing an approach for accommodating a generalized lateral range curve. In addition, basic search tactics, which could be used as the basis for future experimentation with search tactic parameters, were to be developed. The validation criteria addressed include:

- **Concept of SAR Operations:** Assumptions about SAR operations and functions modeled were presented to Coast Guard personnel with SAR expertise for a determination of whether the conceptual implementation is correct.
- **Analytical:** The model was executed to demonstrate appropriate resource and target kinematics. The "reasonableness" of the model's results, in comparison to known POD and coverage factor curves, was demonstrated.

3.1 Model Development

Two distinct scenarios were created to address modeling objectives and meet the above validation criteria. The first scenario, called the POD versus Coverage Factor Scenario, simulated a hypothetical situation that can be easily analyzed and compared with known results from the theory of search literature. The second scenario, called the Parallel Single Track Search Scenario, exercised the scripting capabilities of MarOpsSim and involved the participation of Coast Guard personnel with SAR expertise in the script development process. In both scenarios, the detection model being used was the "generalization" of the previous LRC detection model described in section 2.

3.2.2 Conceptual Model for Generalized Lateral Range Curves

The GLRC is also expressed in the Logit form identified in equation (1). Coefficients of the λ summation are supplied to the simulation as a table. Two values, λ_a and λ_b, are provided for each combination of target type and length, search platform type, altitude, speed, wind speed, wave height, and visibility. These values are defined as follows:

$$\lambda_a = a_1, \text{ And } \lambda_b = \sum_{i=2}^{n} a_i x_i \,. \tag{3}$$

The λ_a term defines the coefficient to the CPA range and the λ_b term quantifies the additive portion. The example below demonstrates the concept using equation (2).

$$P_D = \frac{1}{1 + e^{-(\lambda_a \times CPA - \lambda_b)}} \tag{4}$$

The values supplied for the GLRC scenarios are listed in **Table 2: GLRC Lambda Table.** The sweep width values are included for a later analysis, which will be discussed in section 4.

Table 2: GLRC Lambda Table

Target Type	Target Length	λ_a	λ_b	Sweep Width	Platform Type	Platform Speed	Platform Altitude	Wind	Wave Height	Visibility
Power Boat	15	-0.963	2.4767	5.31	Helo	90	750	5	1	10
Power Boat	15	-0.963	1.99915	4.4	Helo	90	750	10	2	10
Power Boat	15	-0.963	0.6748	2.26	Helo	90	750	15	3	10
Power Boat	15	-0.963	-1.8903	0.29	Helo	90	750	20	4	10

Presently, the tabular data entry approach is the simplest way to adapt MarOpsSim to implement a generalized LRC. Using this approach, the simulation engine can use the appropriate parameters as the environment changes over the course of a simulation run. However, note that this implementation is specific to a particular Logit expression. Although this may be adequate for many situations, it does not cover them all (non-visual sensors such as radar for example). Section 4 describes alternative methods for extending the tabular approach for LRC detection modeling in MarOpsSim.

3.3 The POD versus Coverage Factor Scenario (PCF)

The PCF Scenario is designed to demonstrate the reasonableness of results derived using MarOpsSim and illustrate how the simulation might be applied to SAR theory.

3.3.1 Validation Analysis

The PCF Scenario consists of the following:

- The experiment was designed to generate a POD versus coverage factor curve, as illustrated in Naval Operations Analysis (Operations Analysis Study Group, 1989). Thus, the model was run at varying coverage factors ranging from 0.1 to 2.0, in increments of 0.1.
- Five stationary detection platforms were located at fixed distances (the track spacing associated with the prescribed coverage factor and sweep width) along a barrier.
- One hundred targets transited through the barrier, with motion perpendicular to the line of detection platforms.
- Detections were modeled by GLRC curves reflecting the capabilities of helicopters flying at an altitude of 750 feet and a speed of 90 knots, but configured as stationary detection platforms.
- Weather conditions, in agreement with the supplied Lambda table, were fixed for the entire experiment.
- Twenty-five iterations were run for each coverage factor and detections recorded in the results database.

- The ratio of targets detected to total targets generated was computed for each iteration.
- The mean, standard deviation and variance were calculated across the 25 iterations.
- The resulting means were plotted, along with related theoretical curves.

The simulated environment allows the changing coverage factors to be modeled as a change in the size of the search area (varying Track Spacing (ts) between sensors) as opposed to introducing more sensors. The locations of the sensors and target for one iteration have been plotted in **Figure 7** to illustrate the experimental setup. The stationary sensors are spaced at distances (ts) required to define a particular coverage factor. The targets transit from an initial position to a final position traveling in a direction perpendicular to the stationary sensors.

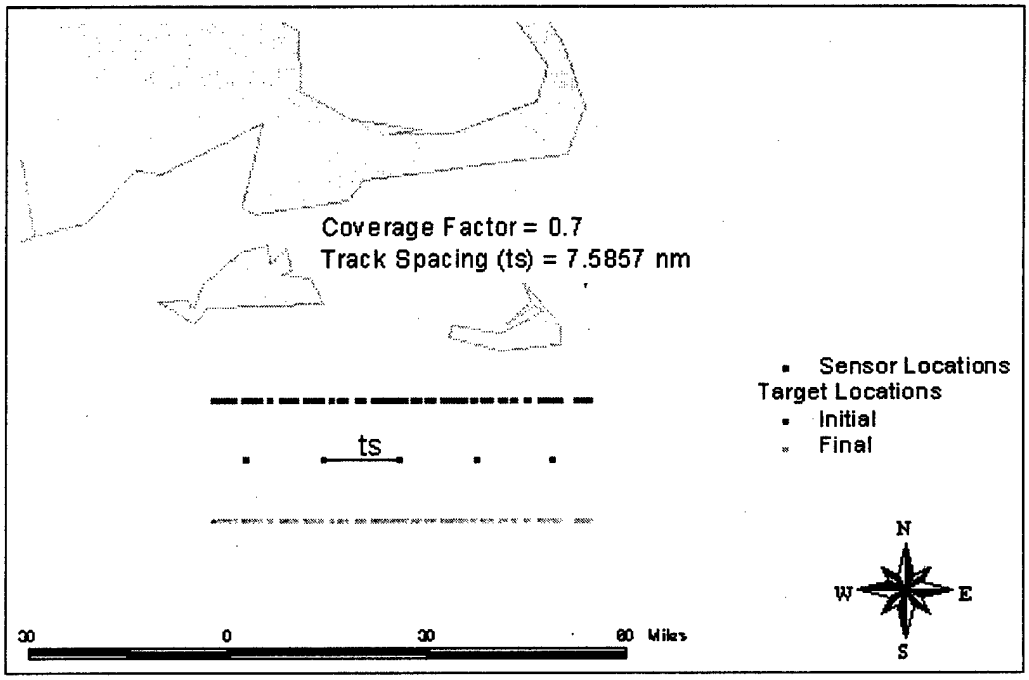

Figure 7: Coverage Factor Scenario Setup.

3.3.2 Results

The results from the experiment were collected and grouped by coverage factor with mean, standard deviation, and variance derived from the iterations performed for each coverage factor. These results are contained in **Table3: POD vs. Coverage Factor Results.**

Table 3: POD vs. Coverage Factor Results

Coverage Factor	Mean POD	Standard Deviation	Variance
0.1	0.1036	0.0350	0.0012
0.2	0.2016	0.0390	0.0015
0.3	0.3004	0.0520	0.0027
0.4	0.4044	0.0600	0.0036
0.5	0.5028	0.0616	0.0038
0.6	0.5996	0.0682	0.0047
0.7	0.6724	0.0675	0.0046
0.8	0.7352	0.0679	0.0046
0.9	0.7796	0.0635	0.0040
1.0	0.8148	0.0552	0.0031
1.1	0.8376	0.0523	0.0027
1.2	0.8708	0.0416	0.0017
1.3	0.8872	0.0370	0.0014
1.4	0.9036	0.0378	0.0014
1.5	0.9188	0.0359	0.0013
1.6	0.9276	0.0360	0.0013
1.7	0.9372	0.0327	0.0011
1.8	0.9440	0.0329	0.0011
1.9	0.9504	0.0308	0.0009
2.0	0.9568	0.0282	0.0008

The data from the table is plotted against the associated theoretical curves in **Figure 8: POD vs. Coverage Factor Curves.** The theoretical curves include representations of the definite range curve, a POD curve analytically derived from the GLRC, the inverse cube curve, and the curve associated with a random search. A discussion of these curves can be found in Search and Screening (Koopman, 1980).

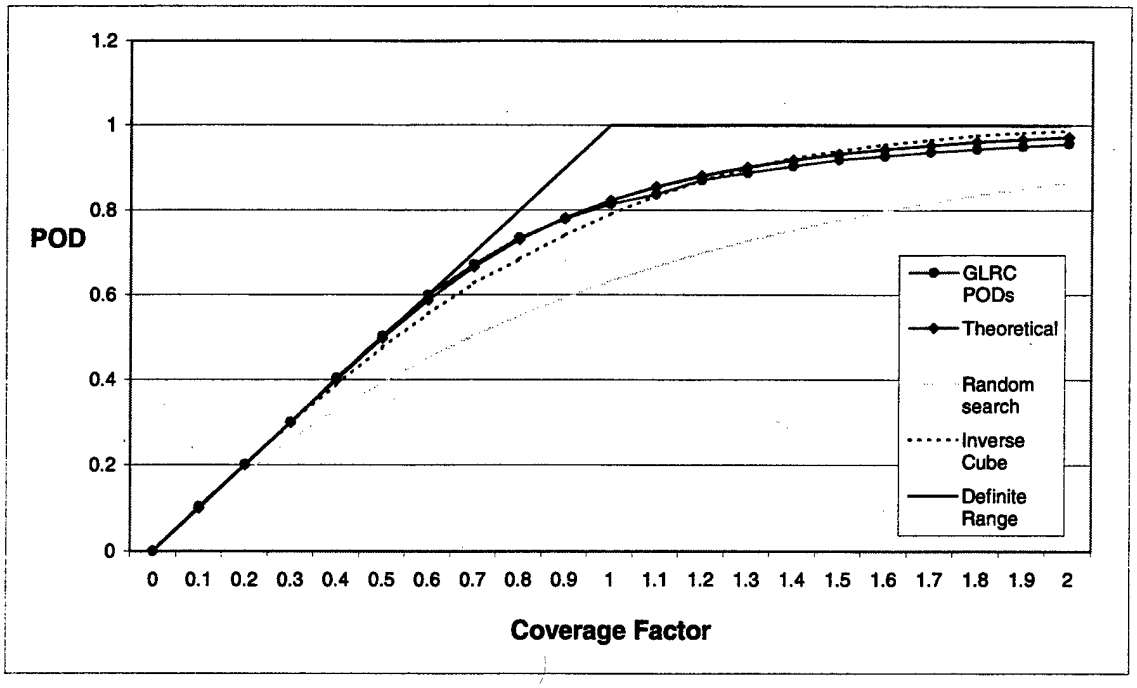

Figure 8: POD vs. Coverage Factor Curves

The analysis of results has been intentionally left simple to allow readers to draw their own conclusions. Further validation of the scenario should be conducted to determine the sensitivity of the results to the target density and number of iterations. Model execution has demonstrated the expected sensitivity to input parameters such as the coefficients of the Logit function and varying coverage factors.

3.4 Parallel Track Single Unit Search Scenario

The PS scenario was designed to exercise the MarOpsSim scripting process and provide enhanced validation of the modeling of target and resource kinematics. The scenario development effort provided a means to explore and demonstrate how the simulation provides the capability to accurately describe SAR tactics and decisions.

3.4.1 Validation Analysis

The PS scenario consists of flying a parallel track search pattern over an area developed about a drifting target. One hundred iterations were run to develop a POD for a coverage factor fixed at 0.7. Model specification includes a SAR planning and coordination unit that receives distress calls from the drifting target, designs a search plan about the target, and dispatches a Search and Rescue Unit (SRU) to the scene to conduct the search. Details of the scenario setup are described in **Appendix C: Parallel Track Single Unit Search Scenario Details. Figure 9** illustrates the model setup, along with a target density plot.

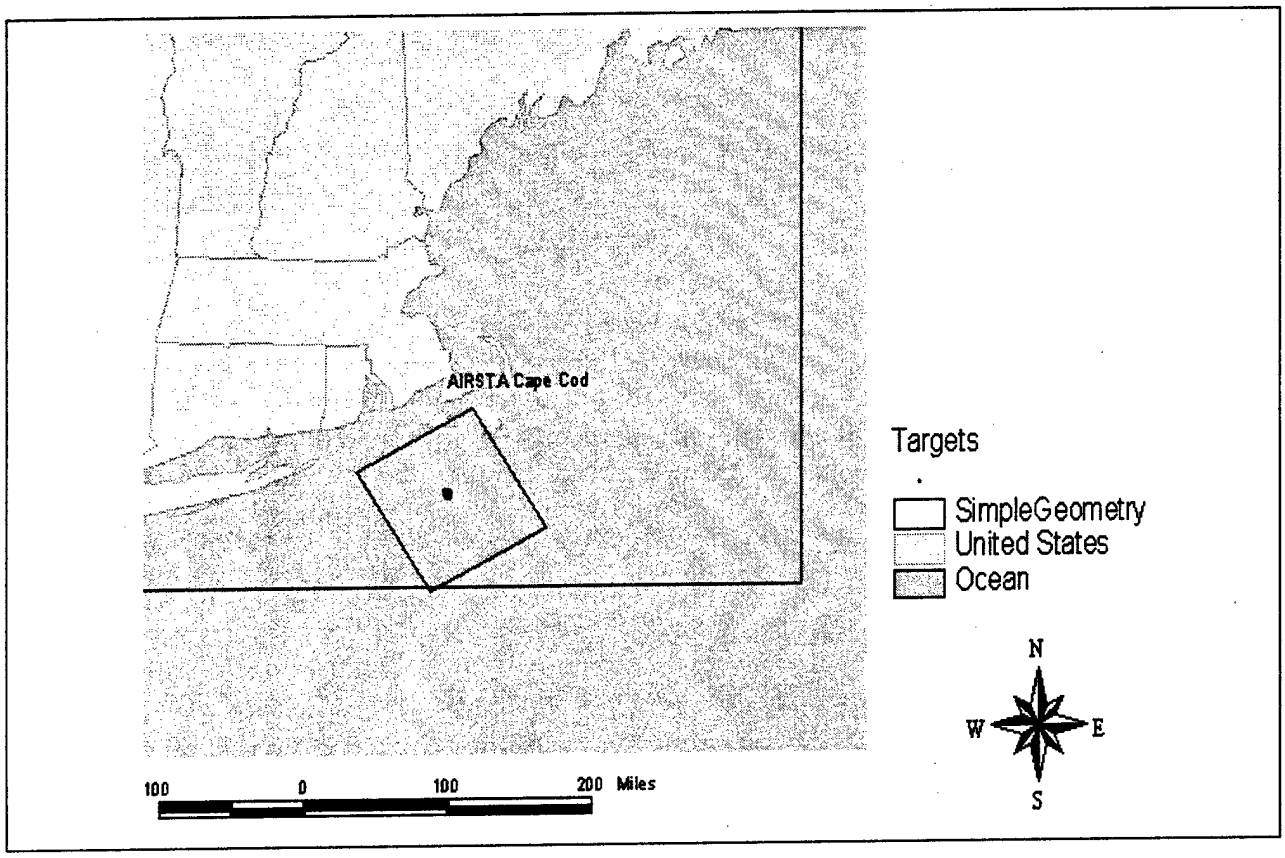

Figure 9: Parallel Search Setup

In Figure 9, the search area is displayed as a black box surrounding the simulated targets. The simple geometry provides the simulation with a hypothetical operational area in which a search mission can be generated and responded to by a Coast Guard platform.

3.4.2 Results

The two MarOpsSim capabilities being evaluated in this scenario are the scenario scripting process and the mechanical output of the simulation.

The scripting process successfully provided a mechanism for accurately describing the SAR planning and response process. The methodology of including operations personnel in the scenario development process proved to be fruitful.

Of the 100 iterations conducted, 79 runs resulted in detections. The high number of detections is most likely due to the clustering of targets along the track line. **Figure 10: Target Distributions** shows the distribution of targets generated over all 100 iterations and illustrates the full search pattern completed whenever the target was not detected. In **Figure 11: A Successful Search** shows iteration 1 where a target is detected. Both figures, generated from MarOpsSim output, portray the behavior and kinematics of the objects modeled in the simulation.

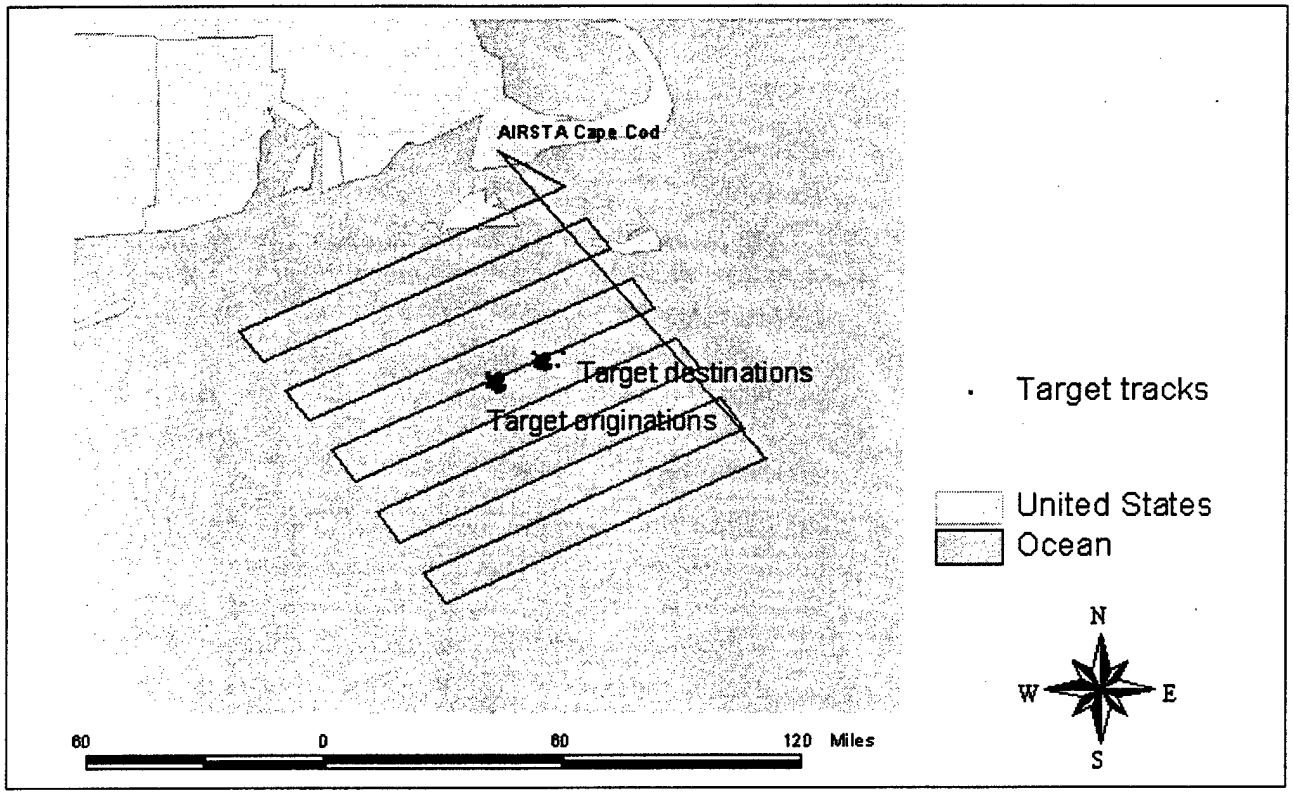

Figure 10: Target Distributions

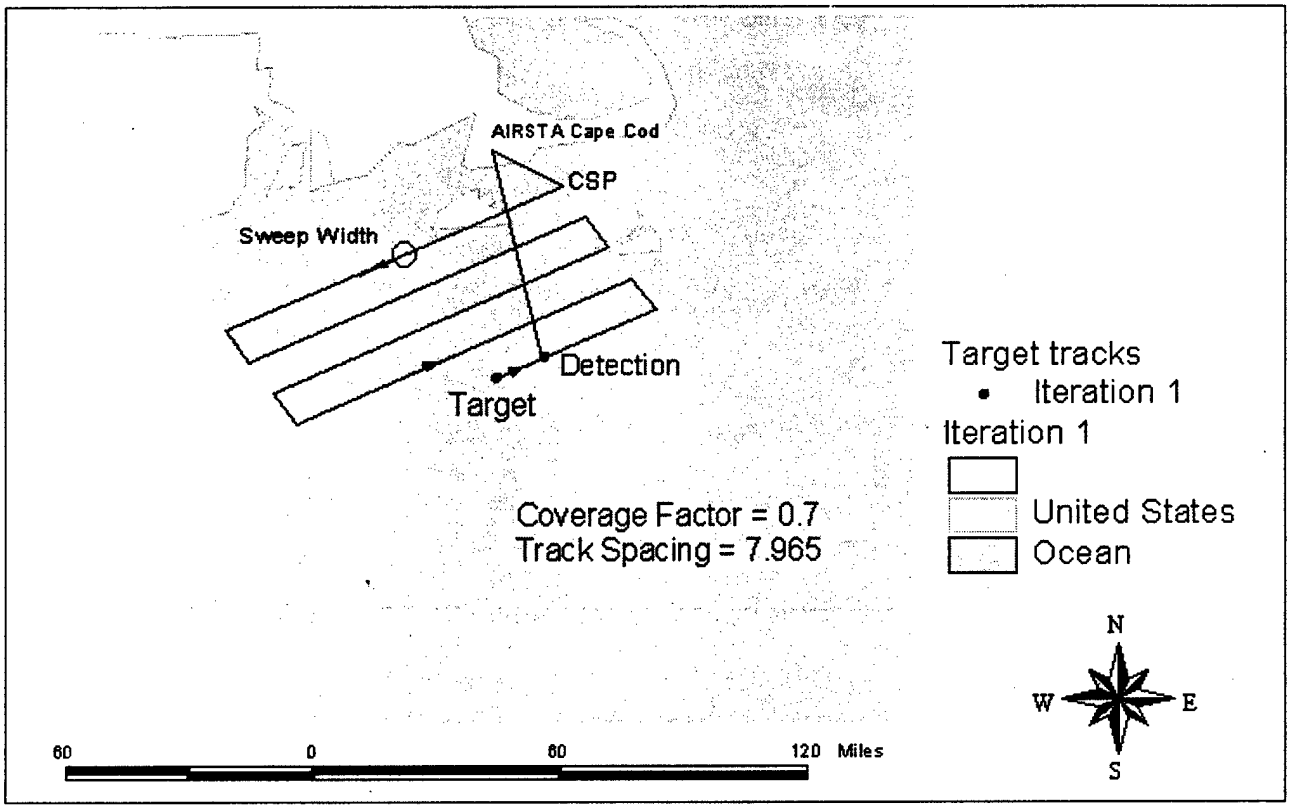

Figure 11: A Successful Search

4. FURTHER SAR APPLICATION

4.1 Model Extensions

The following is a list of potential extensions to the SAR application scenarios that could easily be accomplished:

- The PCF scenario was designed to be consistent with Koopman's derivation of POD versus coverage factor curves that serve as the theoretical basis for similar curves found in the National SAR Manual. The PS and PCF scenarios are very similar as indicated by SAR theory literature. The important difference between the two is that the PS scenario run for this report was biased by the fact that all targets were generated at near zero lateral range from the search platform. By modifying the PS scenario to generate targets uniformly throughout the search area, a comparable analysis of POD versus coverage could be produced. This would enhance model verification and provide a basis for further experimentation.
- The scenarios developed for analysis are simple, yet surprisingly rich in content and could be investigated in more detail. Possible investigations include:
  1. Trade-off analysis comparing search platforms or tactics
  2. Comparison/Validation of LRC models and parameters
  3. Sensitivity analysis on weather and target parameters
- Development of a more comprehensive set of tables to represent other LRC's (e.g. radar or infrared) for comparative studies.
- Development of a library of validated SAR tactics for analysis in the simulation.

4.2 Sensor Modeling Alternatives

Scenarios developed for the SAR application exercises all used the Logit function for detection modeling. Alternative LRC models that the MarOpsSim could include are the inverse cube model, empirical LRCs and the definite range. The first two are discussed in this section. The definite range model would be a special case of the empirical LRC where $P_D = 1$ for CPAs less than or equal to a maximum range.

4.2.1 Empirical Lateral Range Curves

The MarOpsSim model facilitates the empirical definition of an LRC. The actual P_D values at selected CPA ranges may be entered directly into a table instead of the parameters of the Logit formula. This option allows for modeling sensors, such as radar, that do not have experimental results supporting the formulation of a Logit equation.

4.2.2 The Inverse Cube Function

The inverse cube function, as described in Search and Screening (Koopman, 1980), is a visual detection model determined by the following function:

$$\lambda = \frac{kh}{\left(r^2 + h^2\right)^{3/2}} \tag{6}$$

where λ represents the instantaneous detection probability, given the presence of a target, at height of eye (*h*), range (*r*), and conditions (*k*), which are determined by the characteristics of the search platform (e.g., speed), target (size, aspect, and speed), and weather (wave height, precipitation, visibility). The inverse cube model is dependent upon the *k*-value and uniquely determines the LRC. It is therefore critical to supply a valid *k*-value and the corresponding tables or coefficients necessary to correctly define the inverse cube function. The inverse cube function and *k*-values determined from field test data serve as the foundation for the sweep width tables found in the National SAR Manual and Addendum.

A formulation of a second approach to modeling detections based on the inverse cube model can be found in the CASP 2.0 System Specification (Hill, Warne, Hogue & Honec, 1994). This approach derives the probability of detection, P_D, directly from the sweep widths provided in the National SAR Manual (Joint Chiefs of Staff, U.S. Coast Guard, 1991). The formulation is

$$P_D = 1 - \exp\left(\frac{-w^2}{c\pi x^2}\right), \tag{7}$$

where x is the CPA, w is the sweep width, and c is a constant, which equals four in the literature, but is set to 3.333 in practice.

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The analyses conducted in this effort were designed to demonstrate MarOpsSim's utility to SAR studies and provide a high-level validation of the simulation process. The model has proved to be robust and reliable, producing expected or easily explained results. This work also demonstrated MarOpsSim's unique scripting capability that allows it to be tailored to a wide range of potential SAR mission analysis scenarios. The scripts developed as part of this analysis contain reusable components that provide a good foundation for future modeling of the SAR mission.

The results of the scenarios implemented in this task can be summarized as follows:

- The simulated search platform produced detection results consistent with a known LRC.
- A method for generalizing the LRC detection capability was successfully established. Simulation runs conducted with the LRC detection model demonstrated appropriate sensitivity to its driving parameters and produced a reasonable representation of the probability of detection versus coverage factor curve derived from search theory.
- Methods were developed in this effort that allowed for the creation of an accurate representation of the decisions and tactics involved in SAR planning and response.
- The analysis of resource and target motion provided reasonable and expected results.

5.2 Recommendations

Tasks that should follow this effort include:

1. Commissioning an independent verification and validation of the simulation engine, the scripting process and the scenarios developed under this effort.
2. Following the validation effort, begin creating a validated tactics library which will provide the basis for exploring the effectiveness of SAR plans, tactics, capability enhancements and resource allocation schemes.
3. Exploring the scenarios developed for this task in more detail. This includes:
   - Performing sensitivity analysis of the scenario and experimental parameters.
   - Modifying and executing the PS scenario so that its results may be compared to the PCF scenario.
   - Development of data tables for other lateral range curves, such as the inverse cube model and tabular (empirical) LRC's for non-visual sensors.
   - Examining the incorporation of variations on weather and search target motion.
4. Development of an approach for determining the time of detection when using lateral range curves as a detection model. This is expected to be an important factor in some application scenarios where the time of detection can impact scenario results. At the stage of model development documented in this report, MarOpsSim assumes detection times to occur at CPA for LRC detection modeling, which is not realistic.

REFERENCES

1. Edwards, N. C., Jr., Osmer, S. R., Mazour, T. J.& Hover, G. L. (1981) Factors Affecting Coast Guard SAR Unit Visual Detection Performance (CG-D-09-82). Groton, CT: Research & Development Center. (NTIS No. ADA112995)
2. Hill, D. C., Warne, D. M., Hogue, P. K.& Honec III, R. B. (1994) Computer-Assisted Search Planning (CASP) 2.0 System Specification, Naval Surface Warfare Center.
3. Joint Chiefs of Staff & U.S. Coast Guard. (1991) National Search and Rescue Manual, Vol. 1&2, Joint Pub 3-50, COMDNTINST M16120.5A/6A.
4. Koopman, B. O. (1980) Search and Screening, New York: Pergamon Press.
5. Operations Analysis Study Group, U. S. Naval Academy (1989) Naval Operations Analysis, (2nd ed.) Annapolis Maryland: Naval Institute Press.
6. Soza and Company, Ltd. (1996) The Theory of Search - A Simplified Explanation, The Office of Search and Rescue, U.S. Coast Guard.

APPENDIX A: VLRC VALIDATION RUNS

A.1 Scenarios and Model Input

The inputs to MarOpsSim for the VLRC validation runs include parameters and scripts. The scripts include commands to generate targets equally spaced at 0.2-mile intervals. The parameters are listed in succeeding seven tables. **Table A-1: VLRC Global Input Parameters** contains system and configuration level data.

Table A-1: VLRC Global Input Parameters

Globals	Attribute	Value
Globals	Scenario	VLRC
Globals	MaxTime	240
Globals	Iterations	4
Globals	Seed	4,135,793
Globals	Output	Output.txt
Globals	Scripts	Input.txt
Globals	max_events	50,000
Globals	max_simultaneous_events	30,000
Globals	NumResources	1
Globals	NumResourceClasses	1
Globals	NumTargets	25,000
Globals	NumTargetTypes	1
Globals	NumPorts	1
Globals	NumTacticEntries	55

Table A-2: VLRC Input Weather Parameters provide the weather constraints effecting the sensor model.

Table A-2: VLRC Input Weather Parameters

Weather	Attribute	Value
Weather	KValue	2.57
Weather	WaveHeight	2
Weather	WindSpeed	15
Weather	Visibility	10

Table A-3: VLRC Input Geography Parameters defines the limits of the rectangle search area, which provides the constraints for target generation.

Table A-3: VLRC Input Geography Parameters

Geometry	Attribute	Value
Geometry	Min Lat	40
Geometry	Max Lat	40.20
Geometry	Min Lon	-74
Geometry	Max Lon	-73

Table A-4: VLRC Home Ports establishes a generic homeport, labeled "home" from which to initiate the simulation.

Table A-4: VLRC Home Ports

Ports	Latitude	Longitude	Services
Home	0	0	None

Table A-5: VLRC Resource Class Characteristics lists the operating characteristics of the modeled search platform.

Table A-5: VLRC Resource Class Characteristics

Class	Attribute	Value
HH-52A	Amount	1
HH-52A	Sensor	VLRC
HH-52A	Altitude	500
HH-52A	Speed	85
HH-52A	Sweep Width	7.7

Table A-6: VLRC Resource Assignment lists the resources and their availability.

Table B-6: VLRC Resource Assignment

Resource	Class	Name	Homeport
Resources	HH-52A	Searcher	Home

Table A-7: VLRC Target Characteristics lists the operating characteristics of the modeled targets.

Table A-7: VLRC Target Characteristics

Class	Attribute	Value
Generic	Amount	25,000
Generic	Altitude	0
Generic	Speed	0
Generic	IntArrRate	0

A.2 Results and Output Data Analysis

The target spacing produced 79 separate CPAs. The CPAs and associated detection probabilities are listed in three column pairs in **Table A-8: VLRC Resulting P_Ds.**

Table A-8: VLRC Resulting P_Ds

CPA	P_D	CPA	P_D	CPA	P_D
0.0372	0.810127	2.10336	0.371044	4.16526	0.06388
0.11574	0.787184	2.1777	0.342271	4.24386	0.067247
0.19008	0.795095	2.25624	0.363924	4.3182	0.055994
0.26862	0.776108	2.33058	0.320189	4.39674	0.056962
0.34296	0.765032	2.40912	0.33307	4.47108	0.054416
0.4215	0.753956	2.48346	0.291009	4.54962	0.046677
0.49584	0.756329	2.562	0.307753	4.62396	0.048896
0.57438	0.727057	2.63634	0.272082	4.7025	0.050633
0.64872	0.720032	2.71488	0.291139	4.77684	0.041798
0.72732	0.702532	2.78922	0.249211	4.85538	0.045886
0.80166	0.693218	2.86782	0.243671	4.92972	0.041009
0.8802	0.667722	2.94216	0.221609	5.00832	0.045095
0.95454	0.666404	3.0207	0.195411	5.08266	0.031546
1.03308	0.627373	3.09504	0.194795	5.1612	0.045095
1.10742	0.616719	3.17358	0.173259	5.23554	0.024448
1.18596	0.596519	3.24792	0.17429	5.31408	0.033228
1.2603	0.582808	3.32646	0.148734	5.38842	0.02918
1.33884	0.559335	3.4008	0.147476	5.46696	0.030063
1.41318	0.537066	3.4794	0.125	5.5413	0.023659
1.49178	0.515823	3.55374	0.123028	5.6199	0.024525
1.56612	0.490536	3.63228	0.113133	5.69424	0.024448
1.64466	0.464399	3.70662	0.11041	5.77278	0.023734
1.719	0.444795	3.78516	0.106013	5.84712	0.024448
1.79754	0.43038	3.8595	0.087539	5.92566	0.022943
1.87188	0.407729	3.93804	0.093354	6	0.021293
1.95042	0.405854	4.01238	0.070189		
2.02476	0.367508	4.09092	0.078323		

APPENDIX B: POD VERSUS COVERAGE FACTOR SCENARIO DETAILS

This appendix defines the GLRC scenarios at two levels: scenario specification and scenario script. The script contains an experiment setup and a scenario definition. The text in each of these sections is formatted either as received from the scenario developers or to be operational within MarOpsSim. The structure and syntax of the scripts are found in the MarOpsSim Consolidated Software Design Document. The POD versus coverage factor scenario is defined in this appendix.

B.1 Scenarios Specifications

Title: POD vs. Coverage Factor

The setup parameters for the following scenario include:

1. The Scenario name is "POD vs. Coverage"
2. 100 repetitions will be conducted
3. Scenario will run for 12 hours.

Scenario involves the following environment:

1. Geography: D1:
   - boundary = District One boundary
2. Weather: single weather source, with the following characteristics (values set to match λ values provided for Logit equations):
   - wind speed = 5 knots
   - visibility = 10 miles
   - wave height = 1
   - Current Heading = Constant 0 °
   - Current Speed = Constant 0 knots

Sensors are modeled in this scenario using the following detection functions:

1. InvCube:
   - Type = Implicit Inverse cube function (from CASP), using Sweep Width table (from SAR Manual)
2. GLRC:
   - Type = Logit function, using table of λ values (supplied separately)
3. DefiniteRange:
   - Type = Empirical, using table of POD vs. Range values (supplied separately)

The following platforms (with the specified attributes) are required:

1. Buoys With Sensor:
   - Sensor = GLRC
   - Cruising Speed = Stationary
2. Power Boat (16 ft):

- Cruising Speed = 8 knots

Specific platforms include the following:

1. Five assets of type: Buoy, called SRU1, SRU2, SRU3, SRU4, and SRU5
   - OPAREA = D1
   - PrimaryTactic = StationaryDetection
2. Target of type: Power Boat, called SAR Target
   - amount ≈100
   - Interarrival rate = .12 hours
   - Behavior (PrimaryTactic) = Barrier Transit

Scenario performs following steps:

1. Place 5 detection buoys in a straight-line, equal distance apart, forming a barrier across a channel of length L. The first and last buoy positions will correspond with begin and end points of the channel.
2. The targets will be uniformly generated along a line parallel to, and north of the buoys at a distance equal to: track spacing = coverage factor * sweep width
3. The targets will transit north to south at a constant speed.
4. Compute the total number of targets detected and compare to the total number of targets created.

Resource Schedules/Target Generation:
Create Barrier: Place 5 identical buoys in a straight line at a distance s (= track spacing) miles apart 50 miles south of Woods Hole, MA. Orientation will be east to west.
Generate Targets: Targets will be generated using an exponentially distributed inter-arrival rate (time between target arrivals) with a mean of 15 minutes.

1. Determine initial location: Start a target at a position parallel to the line of buoys uniformly distributed between the position of the first and last buoy at a distance equal to the maximum detection range of the buoys.

Scenario details:

1. Commence transit in a direction perpendicular to the orientation of the buoy barrier at a constant speed of 8 knots. The direction will be towards the buoy barrier.

StationaryDetection:

1. Activate buoy sensors and record target detections.

Barrier Transit:

1. Set heading to 180.
2. Set speed to 8 knots.
3. Cruise in a due south direction for a distance equal to three sweep widths.
4. Destroy target.

B.2 Scenarios Scripts

B.2.1 Experiment Setup

The POD versus coverage factor experiment is described in the file setup.mos.

```
//Setup.mos
//The setup parameters for the following scenario include:
Configure SimParameters:
{
//1.    The Scenario name is "POD vs. Coverage"
      SetText Scenario = POD vs. Coverage;

//2.    25 repetitions will be conducted
      SetNum num_iterations = 25;

//3.    Scenario will run for 12 hours.
      SetNum total_hours = 12;

      //      coverage factor set globally for easy manipulation between runs
      SetNum Coverage = 0.666666667;
      SetNum TrackSpacing = 7.965;
      SetNum InitLat = 41.0;
      SetNum InitLon = -70.65;
      SetNum BuoyNum = 0;

      SetText DetectionStatus = Detect;
      SetText DetectionStatus = InMotion;

      SetNum Seed = 4135793;
      SetNum MAX_SIMULTANEOUS_EVENTS = 150;
      SetNum MAX_EVENTS = 150;
      SetNum time_increment = 30;
      SetText time_increment_units = seconds;
}
```

```
Configure Input/Output:
{
        SetText Text_Output = output.txt;
}

Configure Scenario:
{
        Include Scenario.File CoverageFactor;
}
###
```

B.2.2 Scenario Definition

B.2.2.1 Scenario Setup

The POD versus coverage factor scenario is described in the file CoverageFactor.mos.

```
//CoverageFactor.mos
//      Scenario involves the following environment:
Find Environment in /Environment;
Configure Environment:
{
        //      Geography: D1:
        Include Geography.file SimpleGeom;
        Include OPAREAS.file Opareas;

        //      Weather: single weather source, with the following characteristics
        //      (to match available λs for Logit equations):
        Include Weather.file SinglePoint;

        //      Ports: AIRSTA Cape Cod:
        Include Ports.file AirStations;
}

Find Objects in /Platforms;
//      Sensors are modeled in this scenario using the following detection functions:
//      1.      InvCube:
//      2.      GLRC:
Configure Sensors:
{
        Include Sensor.file TheoreticalSensorModels;
}

Configure CommsPackages:
{
        Include CommsPackages.file Radios;
}
```

```
//      The following platforms (with the specified attributes) are required:
Configure Platforms:
{
        Include Asset.file SearchPlatforms; //1.    Buoys With Sensor
        Include TargetType.file Boat; //    2.    Power Boat (15-25 feet)
}

//      Specific platforms include the following:
Find Tactics in /Tactics;
Configure Allocation:
{
        //      Place 5 detection buoys in a straight-line, equal distance apart, forming a barrier
        //      across a channel of length L.  The first and last buoy positions will correspond
        //      with begin and end points of the channel.
        //      1.      Asset of type: Buoy, called SRU1
        Allocate Asset SRU1:
        {
                SetText Class = BuoysWithSensor;
                SetText Homeport = AIRSTACapeCod;
                SetText MainTactic = StationaryDetection;
                SetText OPAREA = D1;
                SetNum Available = 0;
        }
        Allocate Asset SRU2:
        {
                SetText Class = BuoysWithSensor;
                SetText Homeport = AIRSTACapeCod;
                SetText MainTactic = StationaryDetection;
                SetText OPAREA = D1;
                SetNum Available = 0;
        }
        Allocate Asset SRU3:
        {
                SetText Class = BuoysWithSensor;
                SetText Homeport = AIRSTACapeCod;
                SetText MainTactic = StationaryDetection;
                SetText OPAREA = D1;
                SetNum Available = 0;
        }
        Allocate Asset SRU4:
        {
                SetText Class = BuoysWithSensor;
                SetText Homeport = AIRSTACapeCod;
                SetText MainTactic = StationaryDetection;
                SetText OPAREA = D1;
```

```
            SetNum Available = 0;
        }
        Allocate Asset SRU5:
        {
            SetText Class = BuoysWithSensor;
            SetText Homeport = AIRSTACapeCod;
            SetText MainTactic = StationaryDetection;
            SetText OPAREA = D1;
            SetNum Available = 0;
        }

        //      3.      Target of type: Power Boat, called SAR Target
        Generate Target SAR_Target:
        {
            SetNum Quantity = 100;
            SetNum InterArrivalRate = .25;
            SetText Class = MedPowerBoat;
            //      Behavior = Barrier Transit
            SetText MainTactic = BarrierTransit; // behavior
            SetText OPAREA = D1;
            SetNum Available = 0;
        }
}

Configure Tactics:
{
        Include Tactic.file BarrierTransit;
        Include Tactic.file StationaryDetection;
}
###
```

B.2.2.2 Scenario Environmental Files

The environmental files for the POD versus coverage factor scenario consist of the files:

- SimpleGeom.mos
- OPAREAS.mos
- AirStations.mos
- SinglePoint.mos

//SimpleGeom.mos

```
//1.    Geography: D1:
//      boundary = District One boundary
Define Boundary B1:
{
        SetNum UpperLat = 48.0;
```

```
        SetNum LowerLat = 40.0;
        SetNum UpperLon = -67.0;
        SetNum LowerLon = -75.0;
}
###

// OPAREAS.mos
Define OPAREA D1:
{
        SetText boundary = "B1";
        SetVal InterdictionRate = 0;
        SetText weather = SinglePoint;
}
###

//AirStations.mos
Ports: AIRSTA Cape Cod:
Define Port AIRSTACapeCod:
{
        SetNum Latitude = 41.60833;
        SetNum Longitude = -70.55833;
        SetText OPAREA = D1;
}
###

//SinglePoint.mos
//Weather: single weather source, with the following characteristics (to match available λs for
Logit equations):
Include Geography.file SimpleGeom;
Define Weather SinglePoint:
{
        SetText boundary = "B1";
        //      wave height = 1
        SetNum WaveHeight = 1;
        //      wind speed = 5 knots
        SetNum WindSpeed = 5;
        //      visibility = 10 miles
        SetNum Visibility = 10;
        //      Current Heading = Constant 0 o
        SetNum CurrentHeading = 0;
        //      Current Speed = Constant 0 knots
        SetNum CurrentSpeed = 0;
}
###
```

B.2.2.3 Scenario Platform Files

The platform files for the POD versus coverage factor scenario consist of the files:

- TheoreticalSensorModels.mos
- Radios.mos
- SearchBuoys.mos
- Boat.mos

```
// TheoreticalSensorModels.mos
//      2.      GLRC:
Define Sensor GLRC:
{
        //      Type = Logit function, using table of l values (supplied separately)
        //      Series of POD vs. CPA curves described by the equation:
        //      pd = 1/1+exp- (lambda(a)*(CPA)+lambda(b))
        //      where lambda(a) and lambda(b) are found in the lambda values table associated
        //      searcher type, speed altitude, target type, wind speed, wave height and visibility
        SetText Type = Logit;
        SetText Source = GLRC;
}
###

//Radios.mos
Define COMMS_PACKAGE VHF-FM:
{
        SetText Type = RADIO;
        SetText FUNCTION TRANSMITTER;
        SetNum FXRANGE = 100;
        SetNum SIGMATIME = .0001;
        SetNum MEANTIME = .001;
        SetNum PFAILURE = .01;
}
###

//SearchBuoys.mos
//The following platforms (with the specified attributes) are required:
//1.    Buoys With Sensor:
Define platform BuoysWithSensor:
{
        SetText Type = ASSET;
        SetText PlatType = BUOY;

        //      Sensor = GLRC
        SetText SENSOR = GLRC;
```

```
        //      Cruising Speed = Stationary
        SetNum BestEconSpeed = 0;
}
###

//Boat.mos
//      Power Boat (15-25 feet):
Define platform MedPowerBoat:
{
        SetText Type = TARGET;
        SetText PlatType = PowerBoat;
        SetText COMMS_PACKAGE = VHF-FM;
        SetNum LENGTH = 16;
        SetNum DRAFT = 10;
        SetNum BEAM = 20;
        SetText Color = ^white;

        // Cruising Speed = 8 knots
        SetNum BestEconSpeed = 8;
}
###
```

B.2.2.4 Scenario Tactic Files

The scenario tactic files for the POD versus coverage factor scenario consist of the files:

- StationaryDetection.mos
- BarrierTransit.mos

```
//StationaryDetection.mos
Define Tactic StationaryDetection:
{
        RespondTo INIT with InitSD;
        RespondTo Detection with SARDetection;
}

Define TacticalFunction InitSD:
{
        DeclareNum var1;

        SetNum Platform.HEADING = 0;
        SetNum Platform.SPEED = 0;
        SetNum Platform.ALTITUDE = 0;

        //      Place 5 identical buoys in a straight line at a distance s miles apart 50 miles south
        //      of Woods Hole, MA.  Orientation will be east to west.
        SetNum Platform.LatPos = Globals.InitLat;
```

```
        DetermineLocation (Globals.InitLat, Globals.InitLon, (Globals.BuoyNum *
                Globals.TrackSpacing), 90, var1, Platform.LonPos);
        SetNum Globals.BuoyNum = Globals.BuoyNum + 1;
        SetText Platform.MISSION = ^Detect;
        Report (^Self, ^Position);
}

Define TacticalFunction SARDetection:
{
        Report (^Event, ^Detection);
}
###
```

//BarrierTransit.mos

```
Define Tactic BarrierTransit:
{
        RespondTo INIT with InitlTransit;
        RespondTo InTransit with ActualTransit;
        RespondTo CompletedTransit with TransitDone;
}

Define TacticalFunction InitlTransit:
{
        //      Determine initial location: Longitude uniformly distributed from first buoy's
        //      longitude and last buoy's longitude. Latitude = buoy latitude + the buoy's
        //      maximum detection range.
        DeclareNum SW, var1, var2, LocLat;

        //      Start 1.5 times north of SweepWidth
        SetNum SW = Globals.TrackSpacing * Globals.Coverage;
        DetermineLocation (Globals.InitLat, Globals.InitLon, (1.5 * SW), 0,
                Platform.LatPos, var1);

        //      Longitude uniformly distributed between  longitude of first and last buoys.
        DetermineLocation (Globals.InitLat, Globals.InitLon, (0.5 * Globals.TrackSpacing), 270,
                LocLat, var1);
        DetermineLocation (Globals.InitLat, Globals.InitLon, (4.5 * Globals.TrackSpacing), 90,
                LocLat, var2);
        SetNum Platform.LonPos = Uniform (var1, var2, Self.Seed);
        SetText Platform.MISSION = ^Dormant;

        //      Targets will be generated using an exponentially distributed arrival rate
        //      (time between target arrivals) with a mean of 15 minutes.
        //      Leave two hours off the end of the day to do transit
        SCHEDULE (^InTransit, Uniform (0, (Globals.Total_Hours - 2), Self.Seed));
}
```

```
Define TacticalFunction ActualTransit:
{
        DeclareNum SW;

        Report (^Self, ^Position);

        //      Set heading to 180.
        //      Commence transit in a direction perpendicular to the orientation of the buoy
        //      barrier at a constant speed of 8 knots.
        //      The direction will be towards the buoy barrier.
        SetNum Platform.HEADING = 180;

        //      Set speed to 8 knots
        SetNum Platform.SPEED = Self.BestSpeed;
        SetNum Platform.ALTITUDE = 0;
        SetText Platform.MISSION = ^InMotion;

        //Travel south until dist = 3 (SweepWidth) is covered
        SetNum SW = Globals.TrackSpacing * Globals.Coverage;
        SCHEDULE (^CompletedTransit, (3 * SW / Platform.BESTSPEED));
}


Define TacticalFunction TransitDone:
{
        Report (^Self, ^Position);
        SetText Platform.MISSION = ^Dormant;
}
###
```

[This page intentionally left blank.]

APPENDIX C: PARALLEL TRACK SINGLE UNIT SCENARIO DETAILS

This appendix defines the parallel track scenario at two levels: scenario specification and scenario script. The structure and syntax of the scripts are found in the MarOpsSim Consolidated Software Design Document. The parallel track scenario is defined in this appendix. (*Note: MarOpsSim behavior is scripted, no tactical behavior is "hard coded" into the MarOpsSim core. Therefore, the scenario behavior can be controlled by simply modifying the scripts and not MarOpsSim. The following scenario is only one representation for analyzing parallel track search.*)

C.1 Scenarios Specifications

Title: Parallel Track Single (PS)

The setup parameters for the following scenario include:

1. The Scenario name is "Parallel Track Single"
2. 100 repetitions will be conducted
3. Scenario will run for 24 hours.

Scenario involves the following environment:

1. Geography: D1:
   - boundary = District One boundary
2. Weather: single weather source, with the following characteristics (to match available λs for Logit equations):
   - wind speed = 5 knots
   - visibility = 10 miles
   - wave height = 1
   - Current Heading = uniform (1, 360)
   - Current Speed = 2.5 knots
3. Ports: AIRSTA Cape Cod:
   - Latitude = 41.60833
   - Longitude = -70.55833

Sensors are modeled in this scenario using the following detection functions:

1. InvCube:
   - Type = Implicit Inverse cube function (from CASP), using Sweep Width table (from SAR Manual)
1. GLRC:
   - Type = Logit function, using table of values (supplied separately)

The following platforms (with the specified attributes) are required:

1. CommandCenter:
   - CommsPackage = VHF-FM
2. HH60J:
   - CommsPackage = VHF-FM
   - Sensor = GLRC
   - max time on scene = 7 hours
   - Cruising Speed = 90 (override 140 to match available GLRC input)
3. Power Boat (15-25 feet):

Specific platforms include the following:

1. Asset of type: CommandCenter, called SAR_Planner
   - HomePort = AIRSTACapeCod
   - OPAREA = D1
   - PrimaryTactic = SAR_Planning and Coordination
2. Asset of type: HH60J, called SRU
   - HomePort = AIRSTACapeCod
   - OPAREA = D1
   - PrimaryTactic = SAR_Standby
3. Target of type: Power Boat, called SAR Target
   - amount = 1
   - Behavior (PrimaryTactic) = Simple Drift

Scenario performs following steps:

1. Generate SAR target,
2. Send distress to SAR_Planner
3. SAR_Planner selects SRU
4. SAR_Planner builds search area based on track spacing, SRU capabilities, weather and target characteristics
5. SAR_Planner selects CSP and transits to search area.
6. SRU transits to search area:
7. SRU commences search:
   - If found quits search and reports success

- Else continues until last leg is complete, and reports failure

Scenario details:
Simple Drift (Generate SAR Target)

1. Determine initial location: within 50 miles of AIRSTACapeCod, generated within a range determined by a circular normal distribution with a variance of 1 mile.
2. Send distress call to SAR_Planner with estimated location of SAR Target
3. Commence drift

Commence drift:

1. Set heading to direction of current
2. Set speed to speed of current

SAR_Planning and Coordination:

1. Await incoming messages, if distress message is received,
   - Retrieve estimated location of SAR Target from message
   - Plan search with estimated location of SAR Target, given track spacing, expected CSP, L (length), W (width), Orientation,
   - Inform SRU to commence search at CSP given L, W, orientation, center, initial heading of search leg, track spacing and creep (direction).

Plan search (Parallel Track Single - Shape (LxW) is determined as follows: ratio L/W = r) providing: L, W, center, CSP, initial heading (orientation) of search leg, track spacing and creep:

1. Select SRU
2. Determine L as follows:
   - L = length of search leg
   - W = total width or sum of all cross-legs.
   - v = speed of searcher
   - t = max time on scene constraint
   - s = desired track spacing
   - search box area = s*t*v = (L + s) (W+s) = (L + s) * (L/r + s)
   - Solve the quadratic L^2 + L*s*(r+1) + r*(s^2 - s*t*v) = 0 for L:
   - L = (-s(r+1) + SQRT(s^2(r+1) ^2-4(r(s^2 - stv))))/2
3. Set W = L/r.
4. Orientation of the search area is in the direction of drift.

5. Determine CSP = corner closest to location of SRU expecting initial heading of search leg and creep (CSP is offset one half-square track space from the outer corner of the larger search area).

SAR Standby:

1. Await notification, when commanded to search:
   - Transit to CSP with Destination = CSP, Next Task = Patrol/Search, & Return Task = SAR Standby, Return Location = Home Port

Transit:
Given: Next Task, Destination, Return Task, Return Location

1. Starting at current location, move to Destination.
2. Perform Next Task given: Return Task, & Return Location

Patrol/Search:
Given: Return Task, Return Location

1. Starting at CSP, repeat next four steps until target is found or max time on scene is reached:
   - leg 1: move along major axis (parallel to direction of assumed target motion - initially towards center) for L miles (this is your track length),
   - leg 2: move along creep (cross-leg), perpendicular to targets motion, initially towards center, and equal to track spacing
   - leg 3: move along major axis with opposite heading as leg 1 for L miles.
   - leg 4: repeat leg 2.
2. Transit to Return Location with Next Task = Return Task, Destination = Return Location, Return Task = None, Return Location = None

C.2 Scenarios Scripts

C.2.1 Experiment Setup

The parallel single track search experiment is described in the file setup.mos.

//Setup.mos

```
//The setup parameters for the following scenario include:
Configure SimParameters:
{
// 1.   The Scenario name is "Parallel Track Single"
       SetText Scenario = Parallel Track Single;

// 2.   100 repetitions will be conducted
       SetNum num_iterations = 100;
```

```
// 3.    Scenario will run for 24 hours.
      SetNum total_hours = 24;

      //      coverage factor set globally for easy manipulation between runs
      SetNum Ratio = 1;
      SetNum Coverage = 0.7;
      SetNum TrackSpacing = 7.585714285714;
      SetText DetectionStatus = SAR;
      SetText DetectionStatus = SARCase;

      SetNum Seed = 4135793;
      SetNum MAX_SIMULTANEOUS_EVENTS = 10;
      SetNum MAX_EVENTS = 50;
      SetNum time_increment = 30;
      SetText time_increment_units = seconds;
}

Configure Input/Output:
{
      SetText Text_Output = newout3.txt;
}

Configure Scenario:
{
      Include Scenario.File ParallelTrackSingle;
}
###
```

C.2.2 Scenario Definition

C.2.2.1 Scenario Setup

The parallel single track search scenario is described in the file ParallelTrackSingle.mos.

```
// ParallelTrackSingle.mos
//      Scenario involves the following environment:
Find Environment in /Environment;
Configure Environment:
{
      //      Geography: D1:
      Include Geography.file SimpleGeom;
      Include OPAREAS.file Opareas;

      //      Weather: single weather source, with the following characteristics
      //      (to match available ls for Logit equations):
      Include Weather.file SinglePoint;
```

```
        //      Ports: AIRSTA Cape Cod:
        Include Ports.file AirStations;
}

//      The following platforms (with the specified attributes) are required:
Find Objects in /Platforms;
Configure Platforms:
{
        //      1.      CommandCenter:
        Include Asset.file COMMAND;
        //      2.      HH60J:
        Include Asset.file CGHelos;
        //      3.      Power Boat (15-25 feet):
        Include TargetType.file Boat;
}

//Sensors are modeled in this scenario using the following detection functions:
//      1.      InvCube:
//      2.      GLRC:
Configure Sensors:
{
        Include Sensor.file TheoreticalSensorModels;
}

Configure CommsPackages:
{
        Include CommsPackages.file Radios;
}

//      Specific platforms include the following:
Find Tactics in /Tactics;
Configure Allocation:
{
        //      1.      Asset of type: CommandCenter, called SAR_Planner
        Allocate Asset SAR_Planner:
        {
                SetText Class = CommandCenter;
                SetText Homeport = AIRSTACapeCod;
                //      PrimaryTactic = SAR_Planning and Coordination
                SetText MainTactic = SAR_Plan_n_Coordinate;
                SetText OPAREA = D1;
                SetNum Available = 0;
        }
        //      2.      Asset of type: HH60J, called SRU
        Allocate Asset SRU:
        {
```

```
            SetText Class = HH-60J;
            SetText Homeport = AIRSTACapeCod;
            SetText MainTactic = SAR_Standby;
            SetText OPAREA = D1;
            SetNum Available = 0;
        }
        //      3.      Target of type: Power Boat, called SAR Target
        Generate Target SAR_Target:
        {
            SetNum Quantity = 1;
            SetNum InterArrivalRate = 0;
            SetText Class = MedPowerBoat;
            //      Behavior = Simple Drift
            SetText MainTactic = SimpleDrift; // behavior
            SetText OPAREA = D1;
            SetNum Available = 0;
        }
}

Configure Tactics:
{
        Include Tactic.file SAR_Planner;
        Include Tactic.file SAR_Standby;
        Include Tactic.file SimpleDrift;
}
###
```

C.2.2.2 Scenario Environmental Files

The environmental files required for the parallel single track scenario are the same as files contained in appendix **C.2.2.2, Scenario Environmental Files**.

C.2.2.3 Scenario Platform Files

The following files, described in **Appendix C.2.2.3, Scenario Platform Files**, are identical to the files used in the POD versus coverage factor scenario:

- TheoreticalSensorModels.mos,
- Radios.mos,
- Boat.mos

The platform files specific to the parallel single track search are:

- CGHelos.mos
- CommandCenter.mos

//CGHelos.mos

The following platforms (with the specified attributes) are required:

```
//      HH60J:
Define platform HH-60J:
{
        SetText Type = ASSET;
        SetText PlatType = Helo;

        //      Sensor = GLRC
        SetText SENSOR = GLRC;
        //      CommsPackage = VHF-FM
        SetText COMMS_PACKAGE = VHF-FM;
        SetNum FUEL_CAPACITY = 1900;
        SetNum MAXIMUM_SPEED = 85;
        SetNum MAXIMUM_CONTS_SPEED = 165;
        SetNum MAX_RANGE = 400;
        SetNum CRUISEALTITUDE = 750;
        //      max time on scene = 7 hours
        SetNum max_time_on_scene = 7;
        //      Cruising Speed = 90 (override 140 to match available GLRC input)
        SetNum BestEconSpeed = 90;
}
###
```

//CommandCenter.mos:

```
Define platform CommandCenter:
{
        SetText Type = ASSET;
        SetText PlatType = Station;
        SetText COMMS_PACKAGE = VHF-FM;
}
###
```

C.2.2.4 Scenario Tactic Files

The scenario tactic files for the parallel single track search consist of the four files:

- SimpleDrift.mos
- PSPattern.mos
- B2_SAR.mos
- SAR_OPCEN.mos

```
//SAR_OPCEN.mos
//      SAR_Planning and Coordination:
Define Tactic SAR_Plan_n_Coordinate:
{
        RespondTo INIT with OPCEN_INIT;
        RespondTo COMMS with OPCEN_Planning;
}

Define TacticalFunction OPCEN_INIT:
{
        DeclareText body;
        //      TrackSpacing defined globally
        DeclareNum Width, Length, Orientation, TargetLocLat, TargetLocLon, Creep, CSPLat,
                CSPLon, NextCornerLat, NextCornerLon, Mindist;
        SetText body = ^NULL;
        SetNum Width = 0;
        SetNum Length = 0;
        SetNum Orientation = 0;
        SetNum TargetLocLat = 0;
        SetNum TargetLocLon = 0;
        SetNum Creep = 0;
        SetNum CSPLat = 0;
        SetNum CSPLon = 0;
        SetNum NextCornerLat = 0;
        SetNum NextCornerLon = 0;
        SetNum Mindist = 0;
        SetText Platform.MISSION = ^Active;
}

Define TacticalFunction OPCEN_Planning:
{
        DeclareText body;
        DeclareNum Width, Length, Orientation, InitDir, TargetLocLat, TargetLocLon, Creep,
                CSPLat, CSPLon, NextCornerLat, NextCornerLon, Distance1, Distance2,
                Distance3, theta, Mindist;

        //      Await incoming messages, if distress message is received,
        GetMessage (body, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);
        If (body = ^Distress):
        {
                //      Perform DecipherDistressMessage;
                //      Retrieve estimated location of SAR Target from message
                DecipherMessage (TargetLocLat, TargetLocLon, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);

                //      Select SRU
                Select_Asset (^HH-60J, ^SRU, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);
```

```
//      Plan search with estimated location of SAR Target, given track spacing,
//      expecting CSP, L (length), W (width), Orientation,
//      Plan search (Parallel Track Single - Shape (LxW) is determined as
//      follows: ratio L/W = r)
//      SetNum ratio = 1; Globally set
//      providing: L, W, orientation, center, CSP, initial heading of search leg,
//      track spacing and creep:
//      Determine L as follows:
//      L = length of search leg
//      W = total width or sum of all cross-legs.
//      v = speed of searcher
//      t = max time on scene constraint

//      s = desired track spacing
//      coverage factor set globally for easy manipulation between runs

//      search box area = s*t*v = (L + s) (W+s) = (L + s) * (L/r + s)
//      Solve the quadratic L^2 + L*s*(r+1) + r*(s^2 - s*t*v) = 0 for L:
//      L = (-s(r+1) + SQRT(s^2(r+1)^2 - 4(r(s^2 - stv))))/2
SetNum Length = -1 * Globals.TrackSpacing * (Globals.Ratio + 1);
SetNum Distance1 = Globals.TrackSpacing * POC.MaxOnScene * POC.BES;
SetNum Distance2 = POW (Globals.TrackSpacing, 2);
SetNum Distance1 = Distance2 - Distance1;
SetNum Distance2 = (4 * Globals.Ratio) * Distance1;
SetNum Distance1 = Globals.Ratio + 1;
SetNum Distance1 = POW ((Globals.TrackSpacing * Distance1), 2);
SetNum Distance1 = SQRT ((Distance1 - Distance2));
SetNum Length = Length + Distance1;
SetNum Length = Length / 2;

//      Set W = L/r.
SetNum Width = Length / Globals.Ratio;

//      Orientation of the search area is in the direction of drift.
SetNum Orientation = Weather.CurrentHeading;

//      Determine CSP =
//      corner closest to location of SRU expecting initial heading
//      of search leg and creep (CSP is offset one square track space from the
//      outer corner of the larger search area).
//      Perform DetermineCSP;
SetNum Distance1 = POW (Length, 2);
SetNum Distance2 = POW (Width, 2);
SetNum Distance1 = Distance1 + Distance2;
SetNum Distance3 = .5 * SQRT (Distance1);
```

```
//      Upper left corner
SetNum theta = Orientation - ATAN (Length, Width);
DetermineLocation (TargetLocLat, TargetLocLon, Distance3, theta, CSPLat,
        CSPLon);
SetVal Distance1 = Dist (POC.CURRENTLATITUDE,
        POC.CURRENTLONGITUDE, CSPLat, CSPLon);
SetVal Creep = Orientation + 90;
SetNum InitDir = 180 + Orientation;

//      Lower left corner
SetNum theta = Orientation + ATAN (Length, Width);
DetermineLocation (TargetLocLat, TargetLocLon, Distance3, theta,
        NextCornerLat, NextCornerLon);
SetVal Distance2 = Dist (POC.CURRENTLATITUDE,
        POC.CURRENTLONGITUDE, NextCornerLat, NextCornerLon);
SetVal Mindist = min (Distance1, Distance2);
if (Mindist = Distance2):
{
        SetVal CSPLat = NextCornerLat;
        SetVal CSPLon = NextCornerLon;
        SetVal Distance1 = Dist (POC.CURRENTLATITUDE,
                POC.CURRENTLONGITUDE, CSPLat, CSPLon);
        SetVal Creep = Orientation - 90;
        SetNum InitDir = 180 + Orientation;
}

//      Lower right corner
SetNum theta = Orientation + ATAN (Length, Width) + 90;
DetermineLocation (TargetLocLat, TargetLocLon, Distance3, theta,
        NextCornerLat, NextCornerLon);
SetVal Distance2 = Dist (POC.CURRENTLATITUDE,
        POC.CURRENTLONGITUDE, NextCornerLat, NextCornerLon);
if (Mindist = Distance2):
{
        SetVal CSPLat = NextCornerLat;
        SetVal CSPLon = NextCornerLon;
        SetVal Distance1 = Dist (POC.CURRENTLATITUDE,
                POC.CURRENTLONGITUDE, CSPLat, CSPLon);
        SetVal Creep = Orientation - 90;
        SetNum InitDir = Orientation;
}
```

```
                //      Upper right corner
                SetNum theta = Orientation - ATAN (Length, Width) - 90;
                DetermineLocation (TargetLocLat, TargetLocLon, Distance3, theta,
                        NextCornerLat, NextCornerLon);
                SetVal Distance2 = Dist (POC.CURRENTLATITUDE,
                        POC.CURRENTLONGITUDE, NextCornerLat, NextCornerLon);
                if (Mindist = Distance2):
                {
                        SetVal CSPLat = NextCornerLat;
                        SetVal CSPLon = NextCornerLon;
                        SetVal Distance1 = Dist (POC.CURRENTLATITUDE,
                                POC.CURRENTLONGITUDE, CSPLat, CSPLon);
                        SetVal Creep = Orientation + 90;
                        SetNum InitDir = Orientation;
                }

                //      Perform BuildSARMessage;
                //      Inform SRU to commence search at CSP given L, W, orientation, center,
                //      initial heading of search leg, track spacing and creep (direction).
                SendMessage (^SAR, 3, ^USESELF, ^NAME, ^SRU, ^VHF-FM);
                BuildMessage (TargetLocLat, TargetLocLon, Length, Width, Orientation,
                        CSPLat, CSPLon);
                BuildMessage (creep, InitDir, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);
        }
}
###
```

//B2_SAR.mos

```
Include Tactic.file PSPattern;
Define Tactic SAR_Standby:
{
        RespondTo INIT with InitB2;
        RespondTo COMMS with Respond;
        RespondTo TransitToOparea with Transit;
}

Define TacticalFunction InitB2:
{
        SetNum Platform.HEADING = 0;
        SetNum Platform.SPEED = 0;
        SetNum Platform.ALTITUDE = 0;
        SetText Platform.MISSION = ^Bravo2;
}
```

```
Define TacticalFunction Respond:
{
        DeclareText body, NextTask, ReturnTask;
        DeclareNum Width, Length, Orientation, InitDir, TargetLocLat, TargetLocLon, DestLat,
                DestLon, ReturnLat, ReturnLon;

        GetMessage (body, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);
        If (body = ^SAR):
        {
                //      Await notification, when commanded to search:
                DecipherMessage (TargetLocLat, TargetLocLon, Length, Width, Orientation,
                        CSPLat, CSPLon);
                DecipherMessage (creep, InitDir, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);

                //      Transit to CSP with Destination = CSP, Next Task = Patrol/Search, &
                //      Return Task = SAR Standby, Return Location = Home Port
                SetNum DestLat = CSPLat;
                SetNum DestLon = CSPLon;
                SetNum ReturnLat = homeport.latitude;
                SetNum ReturnLon = homeport.longitude;
                SetText NextTask = ^SARSearch;
                SetText ReturnTask = ^SAR_Standby;
                Schedule (^TransitToOparea, 0.08333); // Pause/delay: 5 minutes - to get ready
        }
}

Define TacticalFunction Transit:
{
//      Transit:
//      Given: Next Task, Destination, Return Task, Return Location
        DeclareText NextTask, ReturnTask;
        DeclareNum DestLat, DestLon, ReturnLat, ReturnLon, BeginTime;

        //      Starting at current location, move to Destination.
        SetNum Platform.ALTITUDE = Class.CRUISEALTITUDE;
        SetNum Platform.SPEED = Platform.BESTSPEED;
        CalcH&T (Platform.Heading, BeginTime, DestLat, DestLon, Platform.BESTSPEED);
        SetText Platform.MISSION = ^INTRANSIT;

        //      Perform Next Task given: Return Task, & Return Location
        DivertTo  (NextTask, BeginTime);
}
###
```

```
//PSPattern.mos
Define Tactic SARSearch:
{
        RespondTo INIT with PSSearch;
        RespondTo Detection with SARDetection;
        RespondTo CompletedSearch with Transit;
}

//      Patrol/Search:
Define TacticalFunction PSSearch:
{
        //      Required input: TW - TrackSpace, WIDTH, LENGTH, InitDir, CREEP
        //      Local Variables: CYCLES, COUNTER
        DeclareNum CYCLES, COUNTER, TrackSpace, WIDTH, LENGTH, InitDir, CREEP,
                DestLat, DestLon, ReturnLat, ReturnLon;
        DeclareText NextTask, ReturnTask;

        SetNum Platform.ALTITUDE = Class.CRUISEALTITUDE;
        SetNum CYCLES = WIDTH / (2 * Globals.TrackSpacing);
        SetNum Platform.SPEED = Platform.BES;
        SetText Platform.MISSION = ^SAR;
        SetNum COUNTER = 0;

        //      Starting at CSP,
        //      repeat next four steps until target is found or max time on scene is reached:
        LOOP While (COUNTER < CYCLES):
        {
                //      leg 1: move along major axis (parallel to direction of assumed target
                //      motion - initially towards center ) for L miles (this is your track length),
                Report (^self, ^location);
                SetText Platform.MISSION = ^SAR;
                SetNum Platform.HEADING = InitDir;
                SCHEDULE (^ThisTask, (LENGTH / Platform.BES));

                //      leg 2: move along creep (cross-leg), perpendicular to targets motion,
                //      initially towards center, and equal to track spacing
                //      Perform Leg2;
                Report (^self, ^location);
                SetText Platform.MISSION = ^CrossLeg; // Don't detect n cross legs
                SetNum Platform.HEADING = CREEP;
                SCHEDULE (^ThisTask, (Globals.TrackSpacing / Platform.BES));

                //      leg 3: move along major axis with opposite heading as leg 1 for L miles.
                //      Perform Leg3;
                Report (^self, ^location);
                SetText Platform.MISSION = ^SAR;
```

```
                SetNum Platform.HEADING = InitDir + 180;
                SCHEDULE (^ThisTask, (LENGTH / Platform.BES));

                //      leg 4: repeat leg 2.
                //      Perform Leg4;
                Report (^self, ^location);
                SetText Platform.MISSION = ^CrossLeg; // Don't detect n cross legs
                SetNum Platform.HEADING = CREEP;
                SetNum Platform.HEADING = CREEP;
                SCHEDULE (^ThisTask, (Globals.TrackSpacing / Platform.BES));
                SetNum COUNTER = COUNTER + 1;
        }
        //      Transit to Return Location with Next Task = Return Task,  Destination = Return
        //      Location, Return Task = None, Return Location = None
        //      Given: Return Task, Return Location
        SetNum DestLat = ReturnLat;
        SetNum DestLon = ReturnLon;
        SetNum ReturnLat = 0.0;
        SetNum ReturnLon = 0.0;
        SetText NextTask = ReturnTask;
        SetText ReturnTask = ^None;
        SCHEDULE (^CompletedSearch, 0);
}

Define TacticalFunction SARDetection:
{
        DeclareNum DestLat, DestLon, ReturnLat, ReturnLon;
        DeclareText NextTask, ReturnTask;

        Report (^self, ^location);
        Report (^POC, ^location);
        Report (^event, ^detection);

        SetNum DestLat = ReturnLat;
        SetNum DestLon = ReturnLon;
        SetNum ReturnLat = 0.0;
        SetNum ReturnLon = 0.0;
        SetText NextTask = ReturnTask;
        SetText ReturnTask = ^None;
        SCHEDULE (^CompletedSearch, 0);
}
###
```

```
// SimpleDrift.mos
//      Generate SAR Target

Define Tactic SimpleDrift:
{
        RespondTo INIT with InitDrift;
}

Define TacticalFunction InitDrift:
{
        DeclareText tmp, body;
        DeclareNum InitialLat, InitialLon, SinkTime;

        //      Determine initial location: within 50 miles of AIRSTACapeCod, generated within
        //      a range determined by a circular normal distribution with a variance of 1 mile.
        SetNum Self.LatPos = Normal (40.775, 0.01666666666667, Self.Seed);
        SetNum Self.LonPos = Normal (-70.55833, 0.01313351256011, Self.Seed);
        Report (^self, ^location);

        //      Send distress call to SAR_Planner with estimated location of SAR Target
        SendMessage (^Distress, 3, ^USESELF, ^CLASS, ^CommandCenter, ^VHF-FM);
        BuildMessage (40.775, -70.55833, ^n/a, ^n/a, ^n/a, ^n/a, ^n/a);

        //      Commence drift
        SetText Platform.MISSION = ^SARCase;

        //      Set heading to direction of current
        SetNum Self.SPEED = Weather.CurrentSpeed;

        //      Set speed to speed of current
        SetNum Self.HEADING = Weather.CurrentHeading;
}
###
```