Naval Research Laboratory

Washington, DC 20375-5320

UNCLASSIFIED

NRL/FR/5580--95-9788

Technical Documentation of Nodestar

LAWRENCE D. STONE
THOMAS L. CORWIN

Metron, Inc.
Reston, VA

JAMES B. HOFMANN

Advanced Information Technology Branch
Information Technology Division

December 11, 1995

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

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

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 11, 1995	Interim Report

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Technical Documentation of Nodestar	PE - 61153N PR - RR033034K
6. AUTHOR(S)	
Lawrence D. Stone,* Thomas L. Corwin,* and James B. Hofmann	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Naval Research Laboratory Washington, DC 20375-5320	NRL/FR/5580--95-9788
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Office of Naval Research Arlington, VA 22217-5000	

11. SUPPLEMENTARY NOTES

*Metron, Inc.
Reston, VA 22090

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

13. ABSTRACT (*Maximum 200 words*)

This report presents a general framework for the process of multiple-target, multiple-sensor data fusion. With that framework in place, those areas in which the methodology is mature and ripe for implementation and those areas that require further development are identified. One area in which the methodology is well advanced is nonlinear tracking. For that area, a basic engine, Nodestar, which has been developed to perform nonlinear, multiple-target tracking, is described. The version of Nodestar that has been developed for the Spotlight Advanced Technology Demonstration is described. A discussion of extensions to this version of Nodestar is also included.

14. SUBJECT TERMS		15. NUMBER OF PAGES
Tracking Data fusion Multiple-target tracking	Correlation Nonlinear tracking	70
		16. PRICE CODE

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

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

UNCLASSIFIED

UNCLASSIFIED

CONTENTS

EXECUTIVE SUMMARY E-1

1. INTRODUCTION 1

1.1 Definition of Data Fusion 2
1.2 Related Activities 2

2. GENERAL FRAMEWORK FOR DATA FUSION 2

2.1 Data Fusion 2
2.2 Bayesian Inference Model 3
2.3 Separation of Data Association and State Estimation 5

3. BAYESIAN FILTERING 6

3.1 Formulation 7
3.2 Bayesian Prediction and Smoothing 10

4. NODESTAR ENGINE 13

4.1 State Space 14
4.2 Dynamic Resolution 14
4.3 Motion Model 17

5. NODESTAR LIKELIHOOD FUNCTIONS 19

5.1 Bearing Likelihood 19
5.2 Three-Dimensional Bearing Likelihood 22
5.3 Detection / No Detection Likelihood 25
5.4 Frequency Likelihood 30
5.5 Land Avoidance Likelihood 36
5.6 Elliptical Contact Likelihood 36
5.7 ELINT Likelihood 37

6. MULTIPLE TARGET VERSION OF NODESTAR 37

6.1 Nonlinear Data Association 38
6.2 Probabilistic Data Association 40

7. EXTENSIONS OF NODESTAR 41

7.1 Surface Ship Tracking 41
7.2 Shallow Water Active Acoustic Tracking 42
7.3 Incorporation of Subjective Information 47

REFERENCES 48

APPENDIX A—Spherical Motion Model and Grid 51

APPENDIX B—A Summary of Comparison of Linear and Nonlinear Trackers 57

UNCLASSIFIED

EXECUTIVE SUMMARY

Recently the Navy has felt the need for an improved common tactical picture that incorporates information from all sources. These include Navy, Joint, Allied, and Coalition sources as well as sensor information, passive and active, from air, surface, submarine, and space platforms. In most cases, the types of information are significantly complex and nonlinear. As the Navy moves into operating in more littoral regions, the sensor information will take on an even more complex character. It has quickly become apparent that the present methods and approaches to data fusion, which are based on Kalman filtering and its extensions, are inadequate to cope with this problem.

The Nodestar engine, described in this report, is extremely powerful and represents an advance in the state of the art for data fusion and decision support. Additionally, extensions to Nodestar are straightforward. For any observation or response representing a new type of information or sensor, one must simply construct a likelihood function that computes the probability of receiving that response as a function of target state. This usually requires two things: first, a good physical model of the sensor or response process and, second, that the target state space contain the right information for the calculation, e.g., velocity, if one wishes to account for Doppler in a frequency reading. If the state space contains the necessary information, then one has only to add a new module for calculating the likelihood function. Once this is done, new information can be incorporated via the information update function.

We describe the implementation of Nodestar in the Spotlight decision support system. The Spotlight system demonstrated an innovative computer-based approach to managing and developing resource optimization plans for a deep water ASW surveillance post. Extensions to surface ship tracking and active sonar tracking are described as well.

The bulk of this work was performed under Naval Research Laboratory (NRL) contract N00014-91-C-2140. Development of the Nodestar nonlinear discrete tracker was begun at NRL in 1988 as part of the Tactical ASW Battle Management System (TABS) under the Command Decision Support 6.2 program. Nodestar reached its culmination in the multiple-target tracker developed for the Spotlight Advanced Technology Demonstration (ATD). An extension of the algorithm, also documented herein, was created for multistatic active tracking with the Shallow Water Active Tracking System (SWATS). SWATS work has been performed as part of a Phase I SBIR from NAVSEA and continues under Phase II funding. Appendix A summarizes the technical background for modeling the spherical motion. A summary of the performance comparison of the Nodestar nonlinear tracker (Version 1991) with MTST (Maneuvering Target Statistical Tracker) linear tracker appears in Appendix B.

The Spotlight ATD included the development of a multiple-target, nonlinear correlator-tracker (Nodestar) and a Resource Optimization (RO) system for the Integrated Underwater Surveillance System (IUSS). The correlator-tracker and the RO system in Spotlight are integrated through an innovative Operator-Machine Interface (OMI). The RO system is documented in a separate report.

Nodestar was demonstrated as part of the Spotlight ATD test in December 1993 at the Dam Neck Naval Ocean Processing Facility.

The Spotlight version of Nodestar processes line-of-bearing information (including mirror bearings) and posits (ellipses) while incorporating performance prediction information from System for the Prediction of Acoustic Responses (SPARS). It processes negative (detection/no-detection) information and correctly accounts for the presence of land. The state space is six-dimensional with the following components: position (3D), velocity (2D), and target class (1D). Nodestar uses information about target class, expressed in terms of probability distributions, to help associate and fuse frequency information contained in detection reports. This process requires the use of a threat data base that identifies the set of frequencies radiated by each target class. This information in turn is used to update the distribution on the target's class. A similar process is followed to process information about electronic intelligence (ELINT) parameters.

In comparison to the restrictive assumptions inherent in most current Kalman-filter based tracker-correlators, the ones made for Nodestar allow for much more freedom and fidelity in modeling target motion and sensor response and as such represent an advance in the state of the art in data fusion. The unique features of Nodestar were exploited throughout the Spotlight system to perform more efficiently the functions of forecasting, "what if?" analysis, time-late data inclusion, data correction, and the aforementioned Resource Optimization module. As implemented in Spotlight, Nodestar also represents an advance in the state of the art for decision support.

UNCLASSIFIED

TECHNICAL DOCUMENTATION OF NODESTAR

1. INTRODUCTION

In this report, we present a framework in which significantly complex and nonlinear sensor data can be correctly and effectively addressed. We describe a general framework for the process of multiple target, multiple sensor data fusion. With that framework in place, we identify those areas in which the methodology is mature and ripe for implementation and those areas that require further development. One area in which the methodology is well advanced is nonlinear tracking. For that area we describe a basic engine, Nodestar, which has been developed to perform nonlinear, multiple-target tracking. Then we describe the version of Nodestar that has been developed for the Spotlight Advanced Technology Demonstration. We finish with a discussion of extensions to this version of Nodestar.

Using the Nodestar engine one can incorporate new types of information by the process of calculating likelihood functions. Once the likelihood function has been calculated, the data fusion process (association and state estimation) is performed automatically by the Nodestar engine. This makes the Nodestar engine extremely powerful. It effectively reduces the problem of fusing the information from a new type of sensor to a question of modeling the physics of the sensor, a process that is well understood. Use of this procedure removes the mystery and the need for ad hoc procedures from data fusion. It provides a general approach and framework for performing data fusion.

1.1 Definition of Data Fusion

To place our discussion in a familiar context, we use a modified version of the definition of the data fusion process that has been given by the Joint Directors of Laboratories (JDL) Data Fusion Subpanel (see Waltz and Llinas (1990)). Data fusion consists of performing the activities described in the following paragraphs.

Data association is performed on the set of reports (i.e., sensor responses) produced by the sensors. Association is the process of partitioning this set into a set of disjoint subsets. Each subset in this partition contains all reports associated to a single target and only those reports. In some situations, an additional subset is added to contain the reports attributed to false alarms.

State estimation (target tracking) is the process of combining the reports associated with a single target to produce an estimate of the target's state. The target's state can be very general, including position, velocity, classification, and other target attributes. The estimate can be for the target's past, present, or future state. In tracking nomenclature, these estimates correspond to smoothing (past), filtering (present), and prediction (future).

We have combined attribute classification, which is listed as a separate activity in the JDL definition, into the state estimation process because, in our view, attributes are properly part of the target state space.

Manuscript approved November 3, 1994.

1.2 Related Activities

Data association and state estimation are the core activities that constitute data fusion and are required for a data fusion system. If one is in a single target situation (with no false alarms), then the data association part of the problem may be trivial, but it still takes place. The following activities are closely related to and may be integrated into the data fusion process in some systems. These activities are, in a sense, optional for a data fusion system.

Sensor management is the process of using feedback from the state estimation process (which may include estimation of environmental parameters) to select sensor settings and modes of operation that are most effective for the problem.

Assessment is the process of using the tactical picture produced by the state estimation process to provide estimates of the tactical situation. These estimates may include identification of potential problems and target intentions.

Response planning uses the tactical picture and the expressed goals of the decision maker to recommend actions. Allocation of sensor or processing resources to improve detection or localization of targets is an example of response planning.

2. GENERAL FRAMEWORK FOR DATA FUSION

In this chapter, we present a general framework for data association and state estimation. For data association, this framework will be broadly sketched with areas that require further development indicated.

In the case of state estimation, a mathematical theory based on Bayesian probability describes how this should be done. The principal impediment to the application of this theory, heretofore, has been the lack of computers with sufficient memory and speed to implement a numerical approximation of the theory in a satisfactory manner. Commonly available computers have now evolved sufficient capability for the method to be applied to the full range of tracking problems.

2.1 Data Fusion

We have identified the core activities of data fusion as being data association and state estimation. Before we treat these as separate activities, let us back up a moment and view them as part of single activity, inference. Specifically, the core activity of data fusion is using observations from various sources to estimate the number and state of the targets in the universe of interest. The observations contain ambiguities, uncertainties, and errors. The goal of data fusion is to squeeze as much information as possible from these observations.

From the authors' perspective, Bayesian inference provides the best framework in which to perform this inference task. The choice of a Bayesian framework is based more on pragmatic grounds than on philosophical ones. Of all the common methods of inference (see, for example, Pearl (1988)), it provides the best combination of a logical framework and a powerful set of inference tools. In addition, many of the classical types of non-Bayesian inference are equivalent to Bayesian inference procedures with specific priors. This is true for most classical statistical inference (Lindley 1972) and for classical Kalman filtering, too. Many of the heuristic inference systems have internal inconsistencies that make them suspect at best. This is true of most AI systems that attempt to handle uncertainties (see Pearl (1988)).

The Dempster-Shafer belief function theory offers a logical approach to inference in the presence of uncertainty. It is a generalization of the Bayesian probability model that allows probability to be assigned to a subset without requiring it to be assigned to any members of the

UNCLASSIFIED

subset. However, in its present state of development, the Dempster-Shafer theory lacks the powerful and complete set of inference tools that Bayesian theory provides. As an example, there is not a well-defined and natural notion of expectation for belief functions.

The usual criticisms of the Bayesian approach are that it is computationally intensive, and it requires subjective judgments in terms of prior probability distributions. The cogency of the first objection is quickly disappearing as the computational capability of computers continues its rapid increase. The second objection has substantial appeal until one considers the alternative: no prior information is used at all. One example, taken from tracking submarines, will illustrate this point. We have often been in situations where a skeptic will say, "I have no prior knowledge at all, so how can I construct a prior?" In the case of submarine targets, there is some obvious prior knowledge that is useful to include. For example, the submarines are not going to have a speed faster than 50 kt, or whatever the appropriate top speed is. They certainly are not going to be on land. Usually, you know whether they are in the Atlantic or Pacific and in the northern or southern hemisphere. After answering a few questions like these, a rough prior can be obtained. Often a much better prior can be obtained with more thought. Using even a rough prior is better than having your tracker estimate a submarine's speed to be 100 kt!

2.2 Bayesian Inference Model

In very general terms, our inference problem can be stated as follows. Let $\boldsymbol{X}$ be the state space of our targets. The state space can include position (3D), velocity, and any attribute information that is deemed important, e.g., class of ship or even the ship name. We pick an arbitrary starting time and designate it as $t = 0$. For each $t \geq 0$, we wish to know $N(t)$ the number of targets present and for the nth target, we want an estimate of $X_n(t)$, the state of that target at time t. For this discussion, it is more convenient to think of $N(t)$ as a constant N, the number of targets as constant over the time period of interest. Targets that have not appeared in the area of interest can be assigned to a special state until they do.

Define the stochastic process $\boldsymbol{S}$ as follows:

$$S = \{S(t)\ ;\ t \geq 0\} \equiv \{N, X_1(t), \ldots, X_N(t)\ ;\ t \geq 0\}.$$

The prior information about the targets and their "movements" through state space is incorporated into the probability measure defining this process. Thus, $\boldsymbol{S}$ becomes the prior distribution for the Bayesian inference process. Suppose that by time t, we have a set of observations $O(t) = \{Y_1, Y_2, \ldots, Y_{K(t)}\}$. Each observation Y_k can be a discrete event that occurred at some time u where $0 \leq u \leq t$, or a continuous event such as the time series output from a sensor that has occurred over an interval of time. The data fusion problem is now equivalent to computing the posterior process

$$\begin{aligned} \tilde{S} &= \{S(u)\ ;\ u \geq 0 \mid O(t)\} = \{N, X_1(u), \ldots, X_N(u)\ ;\ u \geq 0 \mid O(t)\} \\ &\equiv \{\tilde{S}(u)\ ;\ u \geq 0\} = \{\tilde{N}, \tilde{X}_1(u), \ldots, \tilde{X}_N(u)\ ;\ u \geq 0\}. \end{aligned} \tag{1}$$

The distribution of $\tilde{S}(u) = \{\tilde{N}, \tilde{X}_1(u), \ldots, \tilde{X}_N(u)\ ; u \geq 0\}$ is the posterior distribution on the state of the system at time u. This is the best estimate of $\tilde{S}(u)$ at time u. The posterior distribution contains all of the information about $\tilde{S}(u)$ that we can infer from $O(t)$. From this distribution we can compute means, 86 percent containment regions, approximate ellipses, or whatever summary

information we wish about the number of targets and their estimated states. If $u = t$, this process is called estimation. If $u > t$, it is called forecasting, and if $u < t$, it is called smoothing.

In a sense we have reduced the problem of data fusion to a complicated application of Bayes' rule. We now know what we would like to compute. The question is: how do we compute it? That is the topic of the rest of this report.

2.2.1 Simplifying Structures

To have any reasonable hope of computing the distribution of $\tilde{S}$, we must make some simplifying assumptions or approximations. One typical approximation is to split the process of computing $\tilde{S}$ into two steps, data association and state estimation. Let us first consider the unified activity for a while longer. Let us assume that the observations in $O(t)$ are obtained at a set of discrete times $0 \le t_1 < t_2 < \ldots < t_k \le t$. Let

$$p(s_1,\ldots,s_k,s) = Pr\{S(t_1) = s_1,\ldots,S(t_k) = s_k, S(t) = s\}$$

be the prior probability (density) that the process S passes through the states $s_1,\ldots,s_k$ at times $t_1,\ldots,t_k$. Suppose that the joint likelihood function L for this set of observations depends only on the system states, $S(t_1),\ldots,S(t_k)$, at these times. Then we have

$$\begin{aligned} Pr\{Y_1(t_1) = y_1,\ldots,Y_k(t_k) = y_k \mid S(u), 0 \le u \le t\} &= Pr\{Y_1(t_1) \\ &= y_1,\ldots,Y_k(t_k) = y_k \mid S(t_1) = s_1,\ldots,S(t_k) = s_k\}. \end{aligned}$$

We can now define

$$L(y_1,\ldots,y_k \mid s_1,\ldots,s_k) \equiv Pr\{Y_1(t_1) = y_1,\ldots,Y_k(t_k) = y_k \mid S(t_1) = s_1,\ldots,S(t_k) = s_k\}.$$

Let $q(t,s) \equiv Pr\{\tilde{S}(t) = s\}$. The function $q(t,\cdot)$ gives the posterior distribution on $S(t)$ given $O(t)$. By Bayes theorem,

$$\begin{aligned} q(t,s) &= \frac{Pr\{Y(t_1) = y_1,\ldots,Y(t_k) = y_k \text{ and } S(t) = s\}}{Pr\{Y(t_1) = y_1,\ldots,Y(t_k) = y_k\}} \\ &= \frac{\int L(y_1,\ldots,y_k \mid s_1,\ldots,s_k)p(s_1,s_2,\ldots,s_k,s)ds_1ds_2\cdots ds_k}{\int L(y_1,\ldots,y_k \mid s_1,\ldots,s_k)p(s_1,s_2,\ldots,s_k,s)ds_1ds_2\cdots ds_k ds}. \end{aligned} \tag{2}$$

Equation (2) gives a formula for calculating the joint probability (density) of $\tilde{S}(t) = \{\tilde{N}, \tilde{X}_1(t),\ldots,\tilde{X}_N(t)\}$. In general, there will be no nice method of computing Eq. (2). Instead, one will have to use numerical methods such as Monte Carlo integration to perform the calculations in Eq. (2). Observe that the process of calculating $q(t,s)$ from Eq. (2) does not involve separate steps for association and tracking. If the computations in Eq. (2) can be performed in a feasible numerical manner, all of the complicated questions of multiple hypothesis management are subsumed in that calculation.

UNCLASSIFIED

2.2.2. Recursive Computation

We now make some additional simplifying assumptions that will allow us to perform the calculation in Eq. (2) in a recursive manner. First, the stochastic process S must be Markovian in the state space $I^+ \times X \times \cdots \times X$, where I^+ is the set of positive integers and the product of the X spaces is taken a number of times equal to a reasonable upper bound on the number of possible targets. Second, the likelihood of an observation $Y(t_i)$ given the past must depend only on the present target state, i.e.,

$$Pr\{Y(t_i) = y_i \mid S(t_1) = s_1, \ldots, S(t_i) = s_i\} = Pr\{Y(t_i) = y_i \mid S(t_i) = s_i\} = L_i(y_i \mid s_i),$$

and the distribution of $Y(t_i)$ must be independent of $Y(t_j)$ given $S(t_1) = s_1, \ldots, S(t_k) = s_k$ for $i \neq j$.

Careful choice of the state space is necessary to make these assumptions reasonable.

Let p_0 be the probability (density) function for $S(0)$ and $p_i(s_i, s_{i-1}) = Pr\{S(t_i) = s_i \mid S(t_{i-1}) = s_{i-1}\}$. Then

$$p(s_1, \ldots, s_k) = \int \prod_{i=1}^{k} p_i(s_i, s_{i-1}) p_0(s_0) ds_0,$$

$$L(y_1, \ldots, y_k \mid s_1, \ldots, s_k) = \prod_{i=1}^{k} L_i(y_i \mid s_i). \tag{3}$$

Substituting Eq. (3) into Eq. (2), we obtain

$$\begin{aligned} q(t_k, s) &= \frac{1}{C'} \int L_k(y_k \mid s) \prod_{i=1}^{k-1} L_i(y_i \mid s_i) p_k(s, s_{k-1}) \prod_{i=1}^{k-1} p_i(s_i, s_{i-1}) p_0(s_0) ds_0 ds_1 \cdots ds_{k-2} ds_{k-1} \\ &= \frac{1}{C'} L_k(y_k \mid s) \int p_k(s, s_{k-1}) \left[\int \prod_{i=1}^{k-1} L_i(y_i \mid s_i) p_i(s_i, s_{i-1}) p_0(s_0) ds_0 ds_1 \cdots ds_{k-2} \right] ds_{k-1} \\ &= \frac{1}{C} L_k(y_k \mid s) \int p_k(s, s_{k-1}) q(t_{k-1}, s_{k-1}) ds_{k-1}, \end{aligned} \tag{4}$$

where

$$C = \int \left[L_k(y_k \mid s) \int p_k(s, s_{k-1}) q(t_{k-1}, s_{k-1}) ds_{k-1} \right] ds.$$

Equation (4) provides a recursive method of computing $q(t_k, s)$. The integral on the right-hand side of the last equality in Eq. (4) performs the "motion update" from the time t_{k-1} of the last computation to the present time t_k. The multiplication by $L_k(y_k|s)$ performs the information update. Division by C renormalizes the numerator to a probability distribution. If there has been no observation at time t_k, then $L_k(y_k|s) = 1$, and there is no information update, only a motion update.

2.3 Separation of Data Association and State Estimation

Most data fusion systems separate the activities of data association and state estimation. From this point on, this report returns to that perspective. The general approach is to first perform the association by developing some method of partitioning the set of observations into a set of disjoint subsets that cover the entire set of observations. The observations in a subset are associated to a single target. No other observations are associated with that target. Having performed the associations, the data fusion problem is reduced to a set of estimation or tracking problems, one for each subset.

The process usually proceeds recursively. It begins with a set of observations. These are associated to form targets and state estimates (tracks) are produced. When the next set of observations is obtained, these tracks are updated to the time of the observations to provide information to the association process. The choice of how to associate the observations to existing or new tracks will often be ambiguous. Most systems require that a single choice be made. Others consider multiple choices (partitions of the set of observations) and assign a probability to each partition or hypothesis. These systems are called multiple-hypotheses correlator-trackers. The term multiple-hypotheses refers to association hypotheses. Even though the processes of association and state estimation take place in separate steps, they are very closely related. In particular, the motion-updated tracker estimates will affect the association of observations to tracks, and obviously the association of observations to tracks affects the state estimates.

The paper by Mori et al. (1986) describes a Bayesian framework for a two-stage process of association and estimation that provides for multiple association hypotheses. Given that one has chosen to view the association and estimation steps as separate although related activities, this paper provides an excellent approach to data association that is far more powerful and flexible than that provided by the Kalman filter-based methods. The interested reader is referred to this paper for the details of the approach. Although Mori et al. do discuss nonlinear state estimation procedures, the discussion and examples given in the paper do not show the full power of this approach nor do they provide a general methodology for implementing a nonlinear state estimation (tracking) process.

The remainder of this report concentrates on describing an approach to nonlinear state estimation and describing a successful and powerful engine, Nodestar, that implements that approach.

3. BAYESIAN FILTERING

We now turn to the problem of state estimation. Following the separation principle described in the previous section, we take the view that the observations have been associated, i.e,. partitioned into subsets with each subset corresponding to a single target. The state estimation problem is to use the observations in a given set to estimate the state of the target. We wish to make a distinction at this point. We call *targets* the actual objects whose state we are estimating. For example, in a surface ship tracking problem, the targets are the ships. A *track*, on the other hand, is the set of observations that are associated into a single subset under the partitioning performed by the association algorithm. The distinction is that targets are real objects. Tracks are constructs created by the data fusion system.

In this section, we describe a state estimation procedure called Bayesian filtering. In a sense, Bayesian filtering is not a new concept. It is very closely related to probabilistic filtering as defined by Jazwinski (1970). This procedure is often mentioned in standard texts and papers on data fusion, such as Bar-Shalom and Fortman (1988), Mori et al. (1986), and Waltz and Llinas (1990). However, it is quickly dismissed as impossible to implement, and the references move on to extended Kalman filtering or weighted Gaussian sum approaches. One purpose of this discussion is to reintroduce this approach and to say that it is feasible and has been implemented in a working system, Nodestar. Nodestar has been tested in operational situations on surveillance problems. Furthermore, Nodestar is not simply a system designed for a single problem. It is an engine that can readily be modified to handle a wide range of tracking problems with diverse data sources.

The fact that Nodestar demonstrates the computational feasibility of implementing a Bayesian filter radically alters the nature of the data fusion problem faced by the R&D community. In the past, the community has spent large amounts of effort trying to adapt Kalman filtering to handle disparate, nonlinear types of information. The inherent limitations of Kalman filtering doom this effort to failure, but researchers have continued to pursue this approach because they have felt that Kalman filtering provided the only computationally feasible approach. Nodestar has changed all this. Bayesian filters have been shown to be computationally feasible and extremely flexible in

UNCLASSIFIED

fusing diverse types of information. Furthermore, the fusion process proceeds according to a consistent and correct mathematical model. In view of this, we feel that researchers should redirect their efforts toward using the Bayesian filtering methodology to develop systems that handle these data fusion problems in a natural, straightforward, and mathematically consistent fashion.

The use of Kalman filtering should be limited to high data rate situations in which the linear-Gaussian assumptions of Kalman filtering are reasonably satisfied. Tracking aircraft with radar is an example of such a situation.

3.1 Formulation

Bayesian filtering is based on the mathematical theory of probabilistic filtering described by Jazwinski (1970). Bayesian filtering is a method of estimating a target's state at a sequence of times. The target's state vector typically includes position but may also include velocity and other variables such as those representing target attributes or characteristics. Let $\boldsymbol{X}$ be the target's state space. The a priori information about the target is represented by a stochastic process $\{X(t);\ t \geq 0\}$, where $X(t)$ is the target state at time t. Sample paths of this process correspond to possible target paths through the state space $\boldsymbol{X}$. For any discrete set of times $0 \leq t_1 < t_2 \ldots < t_{n-1} < t_n$, we define

$$p(x_1,\ldots, x_n) = Pr\{X(t_1) = x_1,\ldots, X(t_n) = x_n\} \tag{5}$$

to be the prior probability (density) of the sample paths of the process that pass through the states $x_1,\ldots, x_n$ at times $t_1,\ldots,t_n$. Suppose that we obtain observations $O(t_n) = \{Y(t_1),\ldots, Y(t_n)\}$ at times $t_1,\ldots,t_n$. In general, these observations are noisy, nonlinear functions of the target state or path. We assume that we can calculate the joint likelihood L for these observations conditioned on target path and that L depends only on the target state at times $t_1,\ldots,t_n$. Thus,

$$Pr\{O(t_n)= (y_1,\ldots, y_n) \mid X(u),\ 0\leq u\leq t_n\} = Pr\{O(t_n) = (y_1,\ldots, y_n) \mid X(t_1),\ldots, X(t_n)\}.$$

Let

$$L(y_1,\ldots, y_n \mid x_1,\ldots, x_n) = Pr\{O(t_n) = (y_1,\ldots, y_n) \mid X(t_1) = x_1,\ldots, X(t_n) = x_n\}. \tag{6}$$

The objective of Bayesian filtering is to calculate

$$q(t_n, x) = Pr\{X(t_n) = x \mid O(t_n) = (y_1,\ldots, y_n)\} \text{ for } x \in \boldsymbol{X}, \tag{7}$$

the posterior distribution on target state at time t_n given the information $O(t_n)$. The function $q(t_n, \cdot)$ is the complete solution to the filtering problem in the sense that it embodies all the information about $X(t_n)$ that is contained in the observations $O(t_n)$ and the prior distribution on $\{X(t);\ t \geq 0\}$. Using Bayes' theorem, we obtain

$$\begin{aligned} q(t_n, x) &= \frac{Pr\{O(t_n) = (y_1,\ldots, y_n) \text{ and } X(t_n) = x\}}{Pr\{O(t_n) = (y_1,\ldots, y_n)\}} \\ &= \frac{\int L(y_1,\ldots, y_n \mid x_1,\ldots, x_{n-1}, x)p(x_1, x_2,\ldots, x_{n-1}, x)dx_1dx_2\cdots dx_{n-1}}{\int L(y_1,\ldots, y_n \mid x_1,\ldots, x_n)p(x_1, x_2,\ldots, x_n)dx_1dx_2\cdots dx_n} \text{ for } x \in S. \end{aligned} \tag{8}$$

In most cases, we are forced to compute $q(t_n, x)$ numerically. The following subsections describe the most common ways of doing this.

3.1.1 Batch Computation

In the batch method, one computes the integrals in Eq. (8) each time a new observation is obtained. Most often a Monte Carlo evaluation of these integrals is performed by replacing the stochastic process $\{X(t);\ t \geq 0\}$ by a finite number M of sample paths drawn from the process. Typically M is about 1000. Usually the paths $(x_1,\ldots, x_n)$ are chosen so that $p(x_1,\ldots, x_n) = 1/M$. Having done this, one computes the likelihood function L for each of these paths as a function of the observations, $O(t_n)$. The numerator in Eq. (8) is the sum of these likelihood functions over all the paths that end at x multiplied by the constant factor $1/M$. The denominator is the same sum taken over all paths. Batch computation is the only method of computing Eq. (8) at this level of generality.

3.1.2 Recursive Computation

When possible, it is more efficient to compute $q(t_n, x)$ in a recursive rather than batch fashion. To do this, we require the following additional assumptions. First, the stochastic process $\{X(t);\ t \geq 0\}$ must be Markovian in the state space $\boldsymbol{X}$. Second, the likelihood of the observation $Y(t_i)$ given the past must depend only on the present target state, i.e.,

$$Pr\{Y(t_i) = y_i \mid X(t_1) = x_1,\ldots, X(t_i) = x_i\} \quad = Pr\{Y(t_i) = y_i \mid X(t_i) = x_i\} \tag{9}$$

$$\equiv L_i(y_i \mid x_i),$$

and the distribution of $Y(t_i)$ must be independent of $Y(t_j)$ given $X(t_1) = x_1,\ldots, X(t_n)= x_n$ for $i \neq j$. Careful choice of the state space usually makes these assumptions reasonable.

Let p_0 be the probability (density) function for $X(0)$ and $p_i(x_i, x_{i-1}) = Pr\{X(t_i) = x_i \mid X(t_{i-1}) = x_{i-1}\}$. Then,

$$p(x_1,\ldots,x_n) = \int \prod_{i=1}^{n} p_i(x_i, x_{i-1}) p_0(x_0) dx_0,$$

$$L(y_1,\ldots,y_n \mid x_1,\ldots,x_n) = \prod_{i=1}^{n} L_i(y_i \mid x_i). \tag{10}$$

Substituting Eq. (10) into Eq. (8), we have

$$\begin{aligned} q(t_n, x) &= \frac{1}{C'} \int L_n(y_n \mid x) \prod_{i=1}^{n-1} L_i(y_i \mid x_i) p_n(x, x_{n-1}) \prod_{i=1}^{n-1} p_i(x_i, x_{i-1}) p_0(x_0) dx_0 dx_1 \cdots dx_{n-2} dx_{n-1} \\ &= \frac{1}{C'} L_n(y_n \mid x) \int p_n(x, x_{n-1}) \left[\int \prod_{i=1}^{n-1} [L_i(y_i \mid x_i) p_i(x_i, x_{i-1})] p_0(x_0) dx_0 dx_1 \cdots dx_{n-2} \right] dx_{n-1} \\ &= \frac{1}{C} L_n(y_n \mid x) \int p_n(x, x_{n-1}) q(t_{n-1}, x_{n-1}) dx_{n-1} \text{ for } x \in X., \end{aligned} \tag{11}$$

where

$$C = \int \left[L_n(y_n \mid x) \int p_n(x, x_{n-1}) q(t_{n-1}, x_{n-1}) dx_{n-1} \right] dx.$$

UNCLASSIFIED

The recursion in Eq. (11) breaks into two steps, motion updating and information updating.

3.1.3 Motion Updating

The integral in the last line of Eq. (11) updates the target state distribution at time t_{n-1} for the "motion" of the target through the state space up to time t_n. Let $q^-(t_n,\cdot)$ denote the motion-updated distribution. Then,

$$q^-(t_n, x) = \int p_n(x, x_{n-1}) q(t_{n-1}, x_{n-1}) dx_{n-1} \text{ for } x \in \boldsymbol{X}. \tag{12}$$

3.1.4 Information Updating

The pointwise multiplication of $q^-(t_n,\cdot)$ by the likelihood function $L_n(y_n, \cdot)$ and the normalization by C constitutes the information update. Thus

$$q(t_n, x) = \frac{1}{C} L_n(y_n | x) q^-(t_n, x) \text{ for } x \in \boldsymbol{X}. \tag{13}$$

Observe that sensor information such as detections, no detections, elliptical position estimates, estimates of the bearing of the target from a sensor, measured signal-to-noise ratio, observed frequency of signal, and many more can be integrated into the tracking solution via the likelihood functions L_n. If there are several types of information or measurements received at a single time, then L_n becomes a joint likelihood function for all the information received.

3.1.5 Nonlinear Tracking

Equations (12) and (13) do not require the observations to be linear functions of the target state. They do not require the measurement errors or the probability distributions on target state to be Gaussian. We call this nonlinear filtering or tracking.

3.1.6 Monte Carlo and Markov Chain Implementations

At this level of generality, Eqs. (12) and (13) must be evaluated numerically. The existence of high-powered scientific workstations allows us to compute and display tracking solutions for complex nonlinear trackers. There are two standard approaches to numerical computation of these equations. In the Monte Carlo approach, the stochastic process $\{X(t); t \geq 0\}$ is approximated by recursively simulating a number M of target paths through the state space $\boldsymbol{S}$. Typically, each path is assigned an a priori probability $1/M$. At each time step t_n, this path is moved forward to its position x_n at time t_n, and its probability is multiplied by $L(y_n \mid x_n)/C$. Moving the paths forward to t_n produces a discrete numerical approximation to Eq. (12). Multiplication by $L(y_n \mid x_n)/C$ accomplishes Eq. (13).

In a Markov chain approach, one typically discretizes the state space so that Eq. (12) is computed through the use of discrete transition probabilities. The likelihood functions are also computed on the discrete state space. Markov chain trackers often operate with more than 300,000 states. To cover a wide range of possible situations, they use a dynamic grid for the state space. The size of the cells in the grid adjusts as the distribution on target state expands or contracts.

3.1.7 Kalman Filters

A Kalman filter tracker represents a special case of the recursion in Eqs. (12) and (13) that holds under special assumptions. The target motion process $\{X(t); t \geq 0\}$ must satisfy a linear stochastic

differential (or difference) equation, and the distribution of $X(0)$ must be Gaussian. The observations $Y(t_i)$ must be linear with Gaussian errors, i.e.,

$$Y(t_i) = M(t_i)x_{t_i} + \varepsilon_i \text{ for } i = 1,\ldots, n, \tag{14}$$

where $M(t_i)$ is a matrix and the ε_i are independent, mean 0, Gaussian random variables for $i = 1,\ldots,n$. Under these assumptions, Jazwinski (1970, Chapter 7) shows that the recursion in Eqs. (12) and (13) reduces to the standard recursive Kalman filter. This is called a linear-Gaussian filter.

The linear-Gaussian restrictions are severe in environments where one is required to fuse diverse types of information. These often involve non-Gaussian distributions and nonlinear relationships between the measurements and the target state. A line of bearing measurement is an example; so is a detection/no detection observation. Many Kalman filters handle this problem by approximating nonlinear measurement relations with linear ones. This is called an extended Kalman filter. The success of this approximation varies. Linear motion models are also restrictive. For example, one cannot have a velocity distribution that varies over space.

3.2 Bayesian Prediction and Smoothing

In filtering terminology (Jazwinski (1970)), *filtering* is the process of estimating the present target state, *prediction* is estimating the target's state at some time in the future, and *smoothing* is estimating the target's state for a time in the past. In this section, we discuss the extension of the above methods to the problems of prediction and smoothing.

In a sense, the prediction problem is a trivial extension of the filtering solution that we have already obtained. One simply applies the motion update step given in Eq. (12) to the function $q(t,\cdot)$, which gives the posterior distribution on target state at the present time t, to compute the distribution on target state at any time $u > t$. For convenience, we consider situations in which there is information about the target's future location that we have not included in the present state estimate. In this case, the problem of prediction becomes more interesting and is parallel to the problem of smoothing.

3.2.1 Formulation

We limit ourselves in this discussion to the situation where the computation can be performed recursively. In practice, this is the only situation in which it is feasible to perform the calculations involved.

Let $[0,T]$ be the time interval of interest, and let $0 \le t_1 < t_2 \ldots < t_{n-1} < t_n \le T$ be a discrete set of times within this interval. Usually, these are times at which observations have taken place. We assume the process $\{X(t);\ t \ge 0\}$ is Markovian and that $O(T) = \{Y(t_1),\ldots, Y(t_n)\}$ is the set of observations obtained in $[0,T]$. Paralleling Eq. (9), we assume that

$$Pr\{Y(t_i) = y_i \mid X(t_1) = x_1,\ldots, X(t_n) = x_n\} = Pr\{Y(t_i) = y_i \mid X(t_i) = x_i\} = L_i(y_i \mid x_i) \tag{15}$$

and that Eq. (10) holds.

For $1 \le k \le n$, define

$$\tilde{q}(t_k, x_k | O(T)) = \frac{Pr\{O(T) = (y_1,\ldots,y_n) \text{ and } X(t_k) = x\}}{Pr\{O(T) = (y_1,\ldots,y_n)\}}. \tag{16}$$

UNCLASSIFIED

Note that $q(t_n, x) = \tilde{q}(t_n, x \mid O(t_n))$. For some constant C'',

$$\tilde{q}(t_k, x_k | O(T)) = \frac{1}{C''} \int \prod_{i=1}^{n} \left[L(y_i | x_i) p_i(x_i, x_{i-1}) \right] p_0(x_0) dx_o \cdots dx_{k-1} dx_{k+1} \cdots dx_n. \tag{17}$$

Observe that we do not integrate over x_k on the right-hand side of Eq. (17). We can write

$$\begin{aligned}
&\tilde{q}(t_k, x_k \mid O(T)) \\
&= \frac{1}{C''} L(y_k \mid x_k) \int p_k(x_k, x_{k-1}) \prod_{i=1}^{k-1} [L(y_i \mid x_i) p_i(x_i, x_{i-1})] p_0(x_0) dx_0 \cdots dx_{k-1} \\
&\qquad \times \int \prod_{i=k+1}^{n} [L(y_i \mid x_i) p_i(x_i, x_{i-1})] dx_{k+1} \cdots dx_n \\
&= \frac{1}{C'''} L(y_k \mid x_k) q^-(t_k, x_k) \int \prod_{i=k+1}^{n} [L(y_i \mid x_i) p_i(x_i, x_{i-1})] dx_{k+1} \cdots dx_n
\end{aligned}$$

where q^- is defined in Eq. (12) and C''' is the appropriate normalizing factor.

Let

$$f_k(x_k | y_{k+1}, \ldots y_n) = \int \prod_{i=k+1}^{n} \left[L(y_i | x_i) p_i(x_i | x_{i-1}) \right] dx_{k+1} \cdots dx_n.$$

Then

$$f_k(x_k \mid y_{k+1}, \ldots y_n) \propto Pr\{X(t_k) = x_k \mid Y(t_{k+1}) = y_{k+1}, \ldots, Y(t_n) = y_n\}.$$

We can now write

$$\tilde{q}(t_k, x_k | O(T)) = \frac{1}{C'''} L(y_k | x_k) q^-(t_k | x_k) f_k(x_k | y_{k+1}, \ldots y_n). \tag{18}$$

Observe that $\tilde{q}$ is proportional to a product of factors corresponding to the effect of past information, $q^-(t_k, x_k)$; the future information, $f_k(x_k \mid (y_{k+1}, \ldots y_n))$; and the present information $L(y_k \mid x_k)$. Specifically, $q^-(t_k, x_k)$ is the probability density that $X(t_k) = x_k$ conditioned on the past information $O(t_{k-1})$. Note that $q^-(t_k, \cdot)$ includes the contact information for times $t < t_k$ and is "motion-updated" from t_{k-1} to t_k. $L(y_k, \cdot)$ is the likelihood function representing the information in the observation $Y(t_k) = y_k$ at the present time, and $f_k(x_k \mid y_{k+1}, \ldots y_n)$ is proportional to the probability density that $X(t_k) = x_k$ given the future observations $Y(t_{k+1}) = y_{k+1}, \ldots, Y(t_n) = y_n$. We call f_k the future likelihood function.

We have already shown how to compute $q^-(t_k, x_k)$ recursively. There is a backward recursion that one can use to compute $f_k(x_k \mid y_{k+1}, \ldots y_n)$. In particular, for $x \in X$,

$$
\begin{aligned}
f_n(x \mid \varnothing) &= 1 \\
f_{n-1}(x \mid y_n) &= \int L(y_n \mid x_n) p_n(x_n, x) dx_n \\
f_k(x \mid y_{k+1}, \ldots, y_n) &= \int f_{k+1}(x \mid y_{k+2}, \ldots, y_n) L(y_{k+1} \mid x_{k+1}) p_{k+1}(x_{k+1}, x) dx_{k+1} \\
&\qquad \text{for } 0 \le k \le n\text{-}2.
\end{aligned}
\tag{19}
$$

3.2.2 Smoothing

The function $\tilde{q}(t,\cdot \mid O(t_n))$ for $0 \le t \le T$ gives the probability distribution representing the smoothed estimate of the target's position at time t given the observations $O(T)$. Let $\mu(t) = \int_X x q(t, x \mid O(T)) dx$ for $0 \le t \le T$. Then $\mu(t)$ is the mean of the smoothed estimate of the target's position at time t. One could plot μ over $[0,T]$ to present an estimated smoothed path for the target given the complete set of observations in $O(T)$.

In the special case where the Kalman filtering assumptions hold, the function $q^-(t_k,\cdot)$ is the normal probability density that results from running a forward Kalman filter on the observations $O(t_{k-1})$ and motion-updating to time t_k. $L(y_k \mid \cdot)$ is a normal density function corresponding to the observation at t_k, and $f_k(x_k \mid y_{k+1}, \ldots y_n)$ is proportional to the output of a backward Kalman filter operating on the contacts $Y(t_{k+1}) = y_{k+1}, \ldots, Y(t_n) = y_n$. One important observation about the presentation given above is that it shows how to compute smoothed tracks for nonlinear filters. Smoothed tracks are usually of interest when one is reconstructing target tracks after the fact.

3.2.3 Land Avoidance

One version of the Nodestar system described below is designed for tracking submarines. We know for certain that submarines will not travel on land. They go around islands and avoid coast lines. Most tracking programs do not account for this piece of information. Kalman filters, for example, are quite capable of projecting submarine tracks across land and even continents. In the submarine surveillance version of Nodestar, we account for this behavior by developing a set of land avoidance likelihood functions that modify the probabilities on the position and velocity components of the target's state as it approaches land. The effect of this is to modify the motion model used by the system. The advantage to doing this by likelihood functions is that these functions can be applied to any underlying motion model. This means that we do not have to reconstruct the motion models to make sure they avoid land each time we change the underlying model. It will happen automatically by use of the land avoidance likelihoods.

The formalism is entirely parallel to that which leads to Eq. (18). For each time t_k, the likelihood function $L(y_k \mid \cdot)$ is the land avoidance likelihood function. Since this function is the same for all times, let us call this function L_a. The value of this function is small for those states close to land and for those velocities that are likely to put the target close to land before a velocity change takes place. With this in mind, Eq. (18) provides a way of computing the distribution on target location at any time t_k knowing that it avoided land in the past, is avoiding it in the present, and will avoid it in the future. Since the land avoidance is known ahead of time, we are actually performing a prediction even though the formalism is that of smoothing.

If an observation is obtained from a sensor at any time t_k, its likelihood function is calculated and combined by pointwise multiplication with the land likelihood function to form a joint likelihood function L_k to be applied at time t_k. With this addition, the computations leading to Eq. (18) are unchanged.

UNCLASSIFIED

Often, the computational load involved in calculating the factor $f_k(x_k \mid y_{k+1},\ldots y_n)$ incorporating the land likelihoods in the future is too burdensome. In this case, we use the recursion in Eqs. (12) and (13) but with each observation likelihood L_n multiplied pointwise by the land avoidance likelihood function.

3.2.4 Modified Motion Model

The application of a land likelihood function L_a to condition the future and present motion of the target is equivalent to modifying the transition probabilities of the Markov motion model. To see this, let the future likelihood function f_n be defined by Eq. (19) for $k = 0,\ldots, n$ with $L(y_k \mid \cdot) = L_a$ for $k = 0,\ldots, n$. Define a modified initial probability measure p_0^* and modified transition probabilities p_i^* for $i = 1,\ldots, n$ as follows:

$$p_0^*(x_0) = \frac{p_0(x_0)f_0(x_0)L_a(x_0)}{\int_X p_0(x')f_0(x')L_a(x')dx'} \quad \text{for } x_0 \in X,$$

$$p_i^*(x_i, x_{i-1}) = \frac{p_i(x_i, x_{i-1})f_i(x_i)L_a(x_i)}{f_{i-1}(x_{i-1})} \quad \text{for } x_{i-1}, x_i \in X. \tag{20}$$

Then, for any path $(x_0,\ldots, x_n)$, we have

$$\begin{aligned} p_0^*(x_0)\prod_{i=1}^{n} p_i^*(x_i, x_{i-1}) &= \frac{p_0^*(x_0)}{f_0(x_0)}\prod_{i=1}^{n} p_i(x_i, x_{i-1})L_a(x_i) \\ &= \frac{1}{C}p_0(x_0)L_a(x_0)\prod_{i=1}^{n} p_i(x_i, x_{i-1})L_a(x_i), \end{aligned} \tag{21}$$

where

$$\begin{aligned} C &= \int_X p_0(x_0)f_0(x_0)L_a(x_0)dx_0 \\ &= \int_X p_0(x_0)L_a(x_0)\left[\int \prod_{i=1}^{n} p_i(x_i, x_{i-1})f_i(x_i)L_a(x_i)dx_1\cdots dx_n\right]dx_0 \\ &= \int_X \cdots \int_X \prod_{i=0}^{n} p_i(x_i, x_{i-1})f_i(x_i)L_a(x_i)dx_0\cdots dx_n. \end{aligned} \tag{22}$$

From Eqs. (21) and (22), we can see that the probability measure on the path $(x_0,\ldots, x_n)$ produced by the modified transition probabilities, p*, is exactly the conditional probability of the path $(x_0,\ldots, x_n)$ given the land likelihood at times $t_0,\ldots, t_n$.

4. NODESTAR ENGINE

Nodestar is a discrete, nonlinear, non-Gaussian, Bayesian filter. It can process disparate forms of information on multiple targets at speeds sufficient for surface ship and submarine applications. Nodestar implements the theory by representing distributions and likelihood functions as values on a discrete grid with dynamically changing resolution. Currently available computers represent most forms of information applicable to surface ship and submarine tracking with appropriate accuracy. The computations required by the theory can be performed with sufficient speed to allow real-time processing.

In the following sections, we describe the Nodestar engine in general and the Spotlight implementation in particular. Even though the emphasis in the following discussion is on the Spotlight implementation, the reader should note that the engine itself is general purpose in that the motion models and likelihood functions described in this section and in Section 5 can be easily replaced by ones that are appropriate for markedly different applications. We provide a hint of that capability in Section 7.

4.1 State Space

Nodestar uses a discrete state space of cells with dynamic resolution. The typical state space has six or more dimensions and 300,000 to 500,000 cells. Each cell contains a probability. The collection of probabilities in the cells is the joint probability distribution for the target state at a given time. In the Spotlight version of Nodestar, the state space is six-dimensional:

Position (3D): The dimensions are latitude, longitude, and depth. The system operates in spherical coordinates so that the latitude-longitude cells at a given time have a fixed width, Δ_{lon} and height Δ_{lat} in latitude and longitude. This means that the cells will be somewhat narrower in the northern parts of the grid than in the southern parts. The motion models and likelihood functions in Nodestar are based on spherical geometry so that all calculations are performed on the sphere. Displays can be presented in any projection desired. The Spotlight version of Nodestar uses a Mercator projection.

Velocity (2D): The spotlight version of Nodestar uses a 2D velocity. This is satisfactory for submarine or surface ship problems. The velocity is the local velocity along a rhumb line course. Appendix A defines a rhumb line.

Attribute: Spotlight uses the class of the submarine target as an attribute. This attribute along with a database specifying the acoustic and radar characteristics of each class allows Nodestar to calculate many of the likelihood functions described below.

One could substitute the ship name or any other attribute for target class if the appropriate database is available to support it. Other attributes can be added. Each additional attribute increases the dimension of the state space, but this is a computational rather than conceptual problem.

4.2 Dynamic Resolution

The resolution of the lat-lon and velocity grids changes dynamically as the size of the probability distribution represented by the grid changes. These dynamic grids are represented by the Nodestar window and velocity wheel.

4.2.1 Spatial Regridding

At each time t, Nodestar calculates the marginal two-dimensional distribution on the lat-lon cells. For each lat-lon cell, it sums the probability from all states whose lat-lon value falls in that cell. For each track (target), there are a fixed number of lat-lon cells, typically 50×50. The location of these cells is determined by a center (lat, lon) and the values of Δ_{lon} and Δ_{lat}. The probabilities in these cells are displayed as color-coded on a monitor. The display takes place in the Nodestar window for that track. This window is superimposed on a geographical display. Usually, $\Delta_{lon} \neq \Delta_{lat}$ so that the window is rectangular rather than square. When regridding, Nodestar can change the length of the cell independently of the width. This is useful if the probability distribution is not square. For example, consider the probability map in Fig. 1.

UNCLASSIFIED

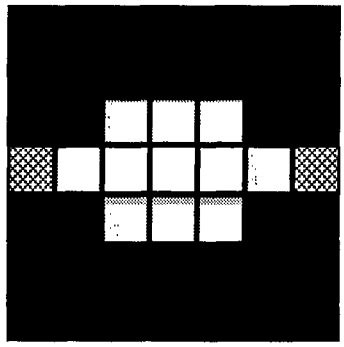


Fig. 1— Grid for spatial probability map

Suppose that we would like to expand the coverage of the probability map in the east-west but not in the north-south direction. For example, suppose that we would like to change our cells so that they look like Fig. 2. In this example, we redefined the cells by doubling the width and keeping the height constant. We call this regridding by the factors (2.0, 1.0). With Nodestar, any pair of positive numbers may be used for regridding factors.

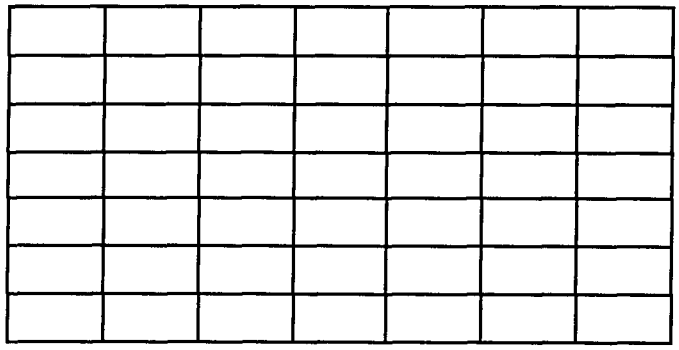


Fig. 2— Grid of new cells

The size and the center of the window are determined dynamically by the location and spread of the marginal distribution on lat-lon in the manner described below. The result is that the center of this window shifts as the center of distribution moves. This size of the window will shrink as the probability distribution becomes more concentrated, but the number of lat-lon cells remains constant. Thus, as the target distribution becomes more localized, the size of the grid cells becomes small and the resolution of the grid becomes finer. As the distribution spreads out, the reverse happens.

The notion of splitting motion updates into steps was developed in response to a problem that Nodestar encounters when the probability distribution is small or the motion update time interval is long. Unless care is taken in these cases, the translation portion of the motion update moves a significant part of the probability mass (or, in extreme cases, all of the mass) outside the Nodestar grid, and this probability is lost.

The solution to this problem is to break the update interval into steps that allow Nodestar to expand the size of the distribution by a specific fraction per step. At the end of each step, Nodestar regrids the distribution so as to create a buffer of empty cells around the distribution that is large enough to accommodate the next motion step. The accuracy and speed of the new method depends heavily on the fraction by which the distribution is allowed to grow. This fraction is one of Nodestar's configuration parameters, the Single Step Motion Threshold. Legal values are between 0.01 and 0.4.

The advantage to the above algorithm is that no mass will be lost because of motion updates. Of course, one does not necessarily want to keep the lowest probability events inside the Nodestar grid, so the spatial regridding algorithm also has a parameter that controls how much mass it may discard before creating the motion buffer. This parameter is called the Max. Mass Lost During Regrid, and legal values are 0.0 to 0.5.

Specifically, the spatial regridding algorithm operates as follows:

0. Set Total Mass Lost Thus Far to zero.

1. Find which of the outermost two rows and two columns has the smallest amount of mass.

2. If this amount of mass plus the Total Mass Lost Thus Far is less than the Max. Mass Lost During Regrid parameter, remove the row or column in question from the Nodestar grid, add the mass in that row or column to the Total Mass Lost Thus Far, and go to step 1.

3. The new Nodestar window is centered in the remaining area, and it is (1.0 + Single Step Motion Threshold) as large as the area remaining after step 2 finishes. This enlargement is performed to handle the next motion step. The new window will have the same number of cells as the original window and the mass in the original grid is distributed to the new grid in proportion to the area of overlap between each of the corresponding cells.

4.2.2 Velocity Regridding

The velocity grid is in the standard (r,θ) planar coordinate system. The coordinate r represents the speed of the target as distance from the center and θ its heading in degrees. The Operator Machine Interface analog of the spatial window is the velocity wheel shown in Figs. 3 and 4. The extent of the wheel is set at the beginning of the problem by specifying a minimum speed r_1 and a maximum speed r_2. The velocity wheel is the annulus of (r,θ) points that lie between r_1 and r_2. As with the spatial grid, the number of velocity cells stays constant over time for a given track (target).

Regridding velocity cells is a more delicate problem than regridding the spatial cells. In the case of the spatial grid, target's position will change slowly, so that Nodestar can move the Nodestar window, keeping the high probability area in the center. If the target starts to move out of the current window area, it will do so slowly enough that Nodestar will be able to shift the window so that it continues to contain the target.

By contrast, the target *can* change its velocity quickly. If Nodestar temporarily abandoned some set of velocities, then it would be quite possible for the target to select one of these when it changes velocity. In this case, Nodestar might lose the track. Nevertheless, it is important to regrid the set of velocities. The way that Nodestar does this is to split just *one* velocity cell at a time, or combine just two cells. In the spatial regridding, all cells are resized at once.

Each velocity cell is created from another by a splitting operation; one can think of the set of velocities as a binary tree with the root being the original velocity cell (containing all conceivable velocities). There are two operations:

1. Nodestar can split a velocity cell into two cells. These cells are called "child" cells, and initially each of them is "childless." Furthermore, they make up a pair of "siblings."

2. Nodestar can combine two childless siblings, making the (common) parent childless.

UNCLASSIFIED

Suppose that there are k velocity cells. Nodestar starts with the original velocity cell (i.e., the annulus) and performs the splitting operation k - 1 times. At each time period, Nodestar calculates the marginal distribution on velocity. This provides a probability for each cell in the wheel. In addition to the fixed number k of velocity cells there is a threshold probability. After each computation of a posterior target state distribution, Nodestar reviews the set of velocities and combines any childless pair of siblings whose total probability does not exceed the threshold. Suppose that Nodestar combines n pairs. It then splits the n most likely cells. Cells can be split radially or angularly. In Fig. 3, Nodestar starts with a velocity wheel and combines two pairs of childless siblings in step 1 and step 2. In Fig. 4, Nodestar splits two other velocity cells.

4.3 Motion Model

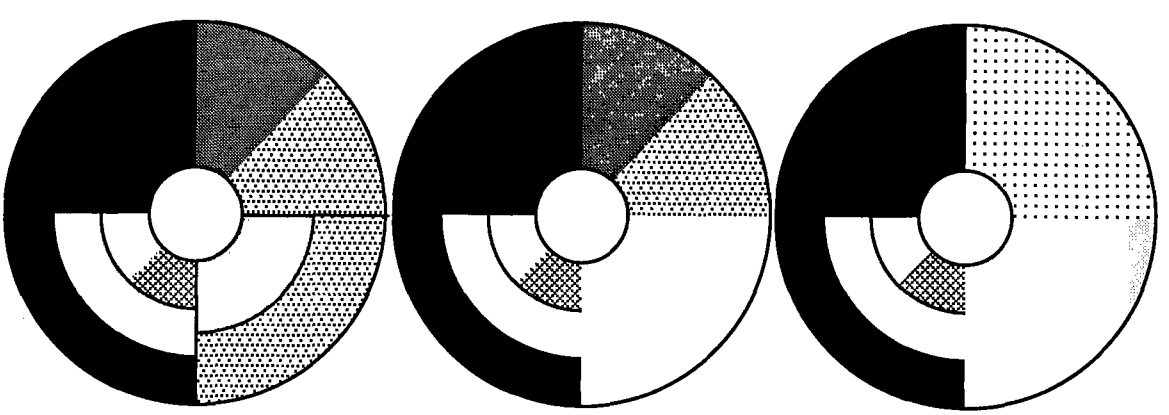
In Nodestar, the target state process $\{X(t);\, t \geq 0\}$ is Markovian with transition function p defined by

$$p_i(x_i, x_{i-1}) = Pr\left\{X(t_i) = x_i \middle| X(t_{i-1}) = x_{i-1}\right\} \text{ for } x_i, x_{i-1} \in \boldsymbol{X}, \text{ and } i = 1, 2. \tag{23}$$

so that

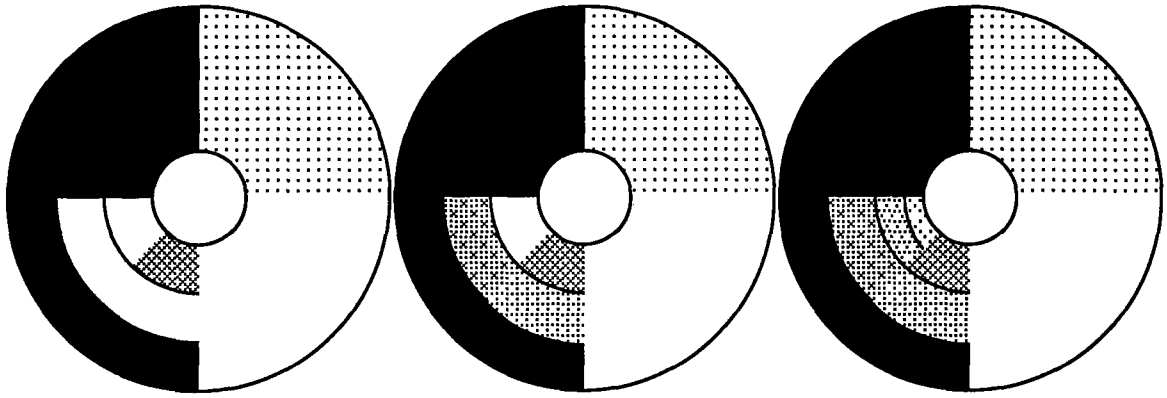
$$p(x_1, \ldots, x_n) = \int_{\boldsymbol{X}} \prod_{i=1}^{n} p_i(x_i, x_{i-1}) p_0(x_0) dx_0 \,. \tag{24}$$

Since the state space is discrete, the integral in Eq. (24) must be interpreted as a summation.



Initial Wheel Step 1 Step 2

Fig. 3 — Combining velocity cells



Initial Wheel Step 1 Step 2

Fig. 4 — Splitting velocity cells

4.3.1 Motion Update

Nodestar uses the recursive method of computing the posterior distribution on target state. Suppose that we have computed $q(t_{n-1},\cdot)$, the posterior distribution on target state at time t_{n-1} . The recursion for computing the posterior distribution on target state at time t_n divides into two steps, a motion update and an information update. We apply the Markov chain motion model to compute $q^-(t_n,\cdot)$, the motion-updated distribution at t_n . If the state space has K cells, then $q(t_{n-1},\cdot)$ becomes a column vector with K components and p_n becomes a K by K matrix of transition probabilities where $p_n(i,j)$ is the probability that the target will transition to cell i at time t_n given it is in cell j at time t_{n-1}. In this case we could, in theory, compute $q^-(t_n,\cdot)$ by matrix multiplication as follows:

$$q^-(t_n,\cdot) = p_n\, q(t_{n-1},\cdot). \tag{25}$$

However, if K is large, this computation becomes very unwieldy. Nodestar uses an alternative method to perform this computation efficiently.

The Nodestar tracker uses an underlying Markov chain target motion model called a generalized random tour. Suppose the target starts at state $x = (z,v,a)$ where z is position, v is velocity, and a is attribute (e.g., target class). It moves with velocity v for a random time τ having an exponential distribution with mean $1/\lambda(x)$. At this point, the target "chooses" a new velocity from a distribution that is a function of $x+v\tau \equiv (z+v\tau, v, a)$ and t. The target continues moving with this new velocity for an independent, exponentially distributed time, and then the process repeats. Nodestar uses a discrete-time version of this process in which a new velocity is drawn at time t_{n-1} with probability, $1-\exp[-\lambda(x)(t_n-t_{n-1})]$, and the velocity is constant during $(t_{n-1}, t_n]$.

To describe the Nodestar computation, we introduce the following notation. Let

$$g_{t_n}(v|z,a) = Pr\{\text{velocity} = v \text{ at time } t_n \mid \text{tgt pos} = z \ \&\ \text{attr} = a \text{ at time } t_n\},$$
$$h_{t_n}(v|z,a) = Pr\{\text{new velocity} = v \mid \text{"choice" is made, tgt pos} = z \ \&\ \text{attr} = a \text{ at time } t_n\},$$
$$r_{t_n}(v|z,a) = Pr\{\text{velocity} = v \text{ during } (t_{n-1},t_n) \mid \text{tgt pos} = z \ \&\ \text{attr} = a \text{ at time } t_{n-1}\}.$$

To obtain $q^-(t_n,\cdot)$, we let $\Delta_n = t_n - t_{n-1}$, and for each (z,v,a) and compute

$$r_{t_n}(v\,|\,z,a) = e^{-\lambda(x)\Delta_n} g_{t_{n-1}}(v\,|\,z,a) + (1-e^{-\lambda(x)\Delta_n}) h_{t_{n-1}}(v\,|\,z,a) \tag{26}$$

UNCLASSIFIED

$$q^{-}\left(t_n,\left(z+\delta\left(t_n,v\right)\right),v,a\right)=r_{t_n}(v|z,a)\sum_{v'}q(t_{n-1},(z,v',a))\,. \tag{27}$$

Note that

$$g_{t_n}(v|\,z,a)=\frac{q(t_n,(z,v,a))}{\sum_{v'}q(t_n,(z,v',a))}\ \text{for}\ (z,v,a)\in \mathrm{X} \tag{28}$$

is computed after the information update at time t_n. The function δ determines whether there is a translation of the probability in cell (z,v,a) to another cell. Suppose that $v = (v_1,v_2)$ and that the spatial grid has cells of width d_i in the ith direction. For each velocity v, the program recursively computes an offset $c_i(v,t_n)$ for the ith dimension. At the beginning, we set $c_i(v,0) = 0$. After the nth motion update, $c_i(v,t_n)$ is incremented by $\Delta_n v_i$. If $c_i(v,t_n) \geq d_i$, then the ith component of $\delta(v,t_n)$ is set equal to kd_i, where k is the largest integer such that $kd_i \leq c_i(v,t_n)$ and $c_i(v,t_n)$ is decremented by kd_i. This process represents an approximate numerical integration of the velocity process to compute the spatial displacement of the target.

Spherical Coordinates. Appendix A discusses how Nodestar adapts this motion model to spherical coordinates.

Information Updating. Nodestar performs its information update through the use of likelihood functions. Having described the state space and motion model used in Nodestar, it remains only to describe the likelihood functions that have been implemented in Nodestar to complete the description of the Nodestar Bayesian filter. The next section describes the likelihood functions that have been implemented for the Spotlight version of Nodestar.

5. NODESTAR LIKELIHOOD FUNCTIONS

In this section, we describe the likelihood functions that have been implemented in the spotlight version of Nodestar.

5.1 Bearing Likelihood

Nodestar uses the following method for computing likelihood functions from bearing observations with time-correlated measurement errors.

5.1.1 Model for Errors in Bearing Observations

Let θ_n and θ_{n-1} be two bearing observations from a sensor at times t_n and t_{n-1}, respectively. Let B_n and B_{n-1} be the actual bearings to the target at times t_n and t_{n-1}, respectively (see Fig. 5.) Note the bearings are calculated in spherical coordinates. Let

$$\varepsilon_n = \theta_n - B_n \quad \text{and} \quad \varepsilon_{n-1} = \theta_{n-1} - B_{n-1}$$

be the errors in the two bearing observations. We assume that the errors $(\varepsilon_n,\varepsilon_{n-1})$ have a bivariate normal distribution. Specifically

$$\varepsilon = \begin{pmatrix} \varepsilon_n \\ \varepsilon_{n-1} \end{pmatrix} \sim N(0,\Sigma)$$

where

$$\Sigma = \begin{pmatrix} \sigma_n^2 & \rho\sigma_n\sigma_{n-1} \\ \rho\sigma_n\sigma_{n-1} & \sigma_{n-1}^2 \end{pmatrix}.$$

In other words, ε_n and ε_{n-1} are normal random variables with mean 0, variance σ_n^2 and σ_{n-1}^2, and correlation ρ.

For calculating bearing likelihood functions in Nodestar, we make the following assumptions:

1. Suppose that we have a sequence of bearing observations $\theta_1, \theta_2, \ldots, \theta_n$ at times $t_1 < t_2 < \ldots < t_n$ with errors $\varepsilon_1, \ldots, \varepsilon_n$. Then the conditional distribution of ε_n given $\varepsilon_1, \ldots, \varepsilon_{n-1}$ depends only on ε_{n-1}, i.e., the bearing errors form a Markov process.

2. The correlation, ρ_n between ε_n and ε_{n-1} depends only on the time difference $t_n - t_{n-1}$. Thus, there is a correlation function c such that $\rho_n = c(t_n - t_{n-1})$.

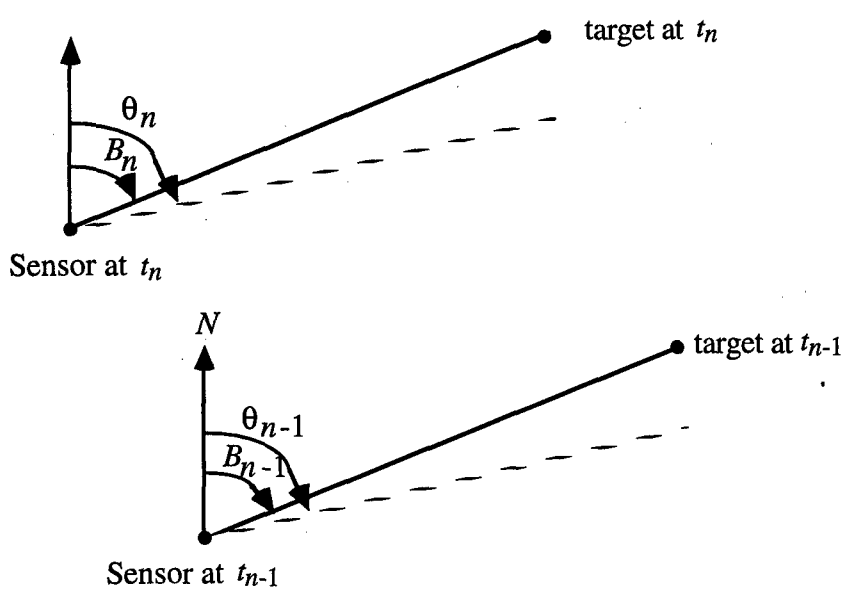


Fig. 5 — Observed and actual bearing to target

Let

$L_B(\varepsilon_n \mid \varepsilon_{n-1}, \ldots, \varepsilon_1) = Pr\{$ obtaining bearing error $\varepsilon_n \mid$ errors $\varepsilon_{n-1}, \ldots, \varepsilon_1\}$.
By assumption 1 we have

$$L_B(\varepsilon_n \mid \varepsilon_{n-1}, \ldots, \varepsilon_1) = L_B(\varepsilon_n \mid \varepsilon_{n-1}). \tag{29}$$

Since $(\varepsilon_n \mid \varepsilon_{n-1})^T$ is bivariate normal with correlation $c(t_n - t_{n-1})$, it follows from Morrison (1976, page 92) that

$$(\varepsilon_n \mid \varepsilon_{n-1}) \sim N(\rho\varepsilon_{n-1}\sigma_n/\sigma_{n-1}, (1-\rho^2)\sigma_n^2)$$

UNCLASSIFIED

where $\rho = c(t_n - t_{n-1})$. Let ϕ be the density function of a unit normal random variable, then

$$L_B(\varepsilon_n \mid \varepsilon_{n-1}) = \phi\left(\frac{\varepsilon_n - \rho\varepsilon_{n-1}\sigma_n/\sigma_{n-1}}{\sigma_n\sqrt{1-\rho^2}}\right). \tag{30}$$

5.1.2 Application to Nodestar

Let $L_B(\varepsilon_1)$ be the likelihood function for the bearing error ε_1. Then

$$L_B(\varepsilon_1) \sim N(0,\sigma_1^2).$$

By using Eq. (29), we may write the joint probability $L_B(\varepsilon_n \mid \varepsilon_{n-1}, \ldots, \varepsilon_1)$ of obtaining errors $\varepsilon_1, \ldots, \varepsilon_n$ as

$$L_B(\varepsilon_1,\ldots,\varepsilon_n) = L_B(\varepsilon_1)\prod_{i=1}^{n-1} L(\varepsilon_{i+1}|\varepsilon_i). \tag{31}$$

5.1.3 Joint Likelihood Function

Suppose that x_i, $i = 1,\ldots,n$ is the target state at times t_i, $i = 1,\ldots,n$. For Nodestar we assume that the target state is six-dimensional and the motion process is Markovian. Let p be the transition density for that process and p_0 be the distribution on target state at time t_0.

Suppose line of bearing detections are made at times $t_1,t_2,\ldots,t_n$ and that $\theta_1,\ldots, \theta_n$ are the observed bearings. Let $B_i(x)$ be the true bearing from the sensor to the target at time t_i given the target state is x. By Eq. (31) and the Markov assumption on the target motion process, one can write the joint probability (density) of the target being in states $x_0, x_1,\ldots, x_n$ and having bearing errors $\varepsilon_i = \theta_i - B_i(x_i)$ at times $t_1, \ldots, t_n$ as

$$\begin{aligned} &Pr\{(x_0,x_1,\ldots,x_n) \text{ and } (\varepsilon_1,\ldots,\varepsilon_n)\} \\ &= p_0(x_0)\prod_{i=1}^{n} p_i(x_i \mid x_{i-1})L_B(\varepsilon_1)L_B(\varepsilon_1|\varepsilon_{i-1}) \\ &= p_0(x_0)p_1(x_1|x_0)L_B(\varepsilon_1)\prod_{i=2}^{n} p_i(x_i \mid x_{i-1})L_B(\varepsilon_1|\varepsilon_{i-1}). \end{aligned} \tag{32}$$

An important feature of Eq. (32) is that it allows us to recursively update for the line of bearing observations.

5.1.4 Recursive Updating

Let $q^-(t_n,\cdot)$ be the probability distribution on target state obtained from information updating through time t_{n-1} and motion updating through time t_n. Note that the bearing error depends only on target position and not on the other components of target state.

Let x be the target's state at time t_n. Then $x = (z,v,a)$ where z is the target's position (e.g., lat, lon, depth), v is the two-dimensional velocity vector, and a is an attribute. For this calculation, we assume that the target does not change velocity during the interval t_{n-1},t_n. Note: if $\lambda(x)(t_n-t_{n-1})$ is small (e.g., $\lambda(x) = 1/60$ minutes and $t_n-t_{n-1} = 10$ minutes), this is a reasonable assumption. If t_n-t_{n-1} is large, then $c(t_n-t_{n-1})$ becomes small and the assumption does not affect the likelihood calculation.

Compute

$$\Delta_n = t_n - t_{n-1}$$

$$\varepsilon_n = \theta_n - B_n(x) = \theta_n - B_n(z,v,a)$$

$$\varepsilon_{n-1} = \theta_{n-1} - B_{n-1}(x-\Delta_n v) = \theta_{n-1} - B_{n-1}(z-\Delta_n v, v, a)$$

$$\rho = c(\Delta_n).$$

Note that B_n, B_{n-1}, ε_n, and ε_{n-1} depend on x and on the sensor's position at times t_n and t_{n-1}. Also observe that under our constant velocity assumption, $z-\Delta_n v$ is the target's position at time t_{n-1}. Now we can compute

$$l(x) = L_B(\varepsilon_n \mid \varepsilon_{n-1}) = \phi\left(\frac{\varepsilon_n - \rho\varepsilon_{n-1}\sigma_n/\sigma_{n-1}}{\sigma_n\sqrt{1-\rho^2}}\right) \quad \text{for } x \in X \tag{33}$$

where X is the set of possible target states. Finally we update for the bearing observation by multiplying l pointwise by $q^-(t_n,\cdot)$ to obtain

$$l(x)q^-(t_n,x) \quad \text{for } x \in X.$$

The standard deviations σ_n and σ_{n-1} will be supplied as part of the bearing detection report. For the function c, we use

$$c(s) = e^{-s/\tau} \qquad \text{for } s \geq 0$$

where τ is the integration time of the sensor.

5.1.5 Mirror Bearings

Mirror bearings are easily handled in Nodestar. If there is an unresolved mirror bearing at time t_n, that means the observation at time t_n is either θ_n or θ_n^m where θ_n^m is the mirror bearing to θ_n. Let l and l^m be the likelihood functions corresponding to the observations θ_n and θ_n^m, respectively. Then the likelihood function l' for the observation either θ_n or θ_n^m is given by

$$l'(x) = l(x) + l^m(x) \quad \text{for } x \in X. \tag{34}$$

5.2 Three-Dimensional Bearing Likelihood

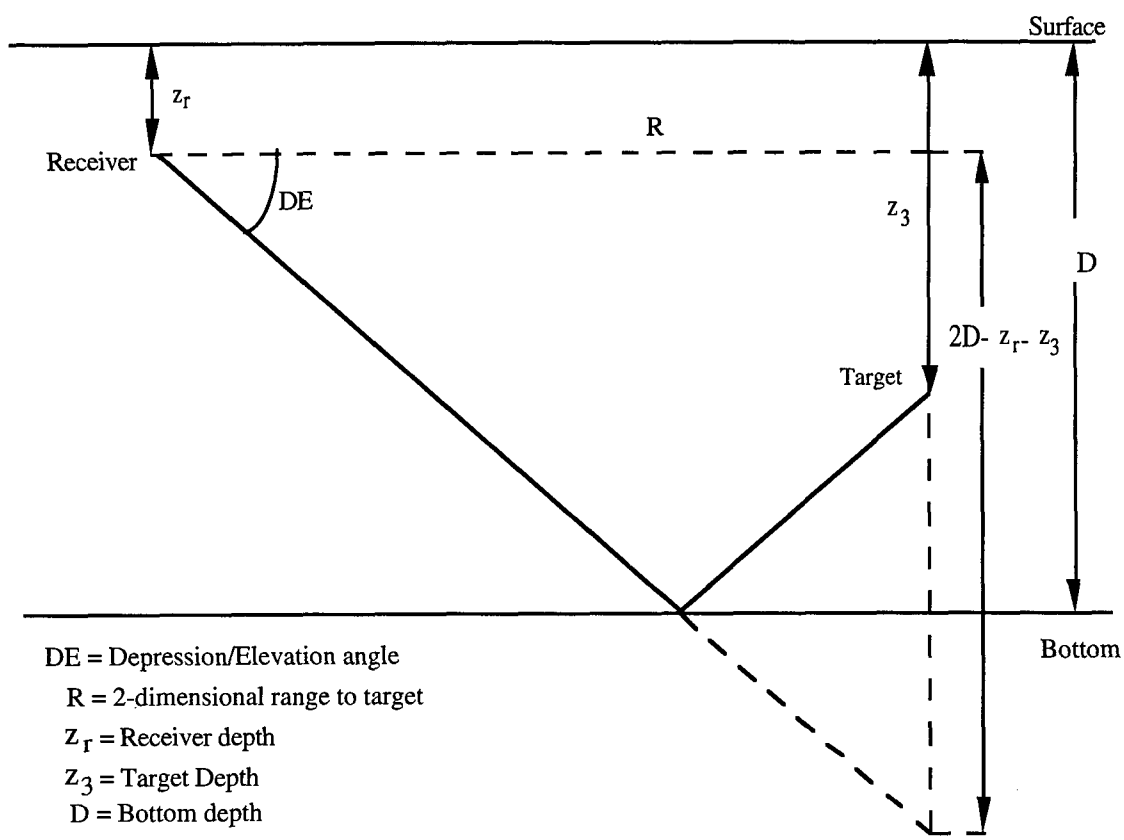
Normally when incorporating a towed array bearing into a Nodestar map, two-dimensional geometry is used to compare the observed bearing with the bearing implied by a given target state. This is the case for the bearing likelihood functions presented above. When the target and array depths are small compared to the ranges involved, this two-dimensional approximation is reasonable. Two exceptions are bearings that result from detection of a surface ship near an array and bottom bounce detections of a submarine by an array. In this section we describe how one can account for three-dimensional bearings in the Nodestar bearing likelihood function.

UNCLASSIFIED

5.2.1 Bottom Bounce Bearings

Figure 6 shows the depression/elevation (DE) angle resulting from a bottom bounce path from a target at range R from the array. We make the idealized assumptions that the sound rays travel in straight lines and that the angle of incidence is equal to the angle of reflection at the ocean bottom. Figure 6 shows how to calculate the DE angle in terms of range, depth of target, depth of the receiver, and ocean depth. The figure also shows that the apparent depth of the target from the array is $d = 2D\text{-}z_r\text{-}z_3$, where D = ocean depth, z_3 = target depth, and z_r = receiver depth.

Consider a target that is at real (or apparent) depth z_3 and whose position projected up to the plane of the array is (z_1,z_2) in the coordinate system shown in Fig. 7. The three-dimensional range and bearing of the target from the array are denoted by r and θ. The two-dimensional bearing of the target in the (z_1, z_2) plane is denoted by β_r. If the towed array detects this target, it will call a bearing $\theta + \varepsilon$ where ε is the error in the observed bearing. If z_3 is small then $\theta \cong \beta_r$. In this case one can simply deal with two-dimensional bearings when computing the likelihood. If z_3 is large, as it will be for the apparent depth d resulting from a bottom bounce detection, then one must account for the DE angle in computing the likelihood. We propose the following method.



$$\mathrm{DE} = \tan^{-1}\left(\frac{2D - z_r - z_3}{R}\right)$$

Fig. 6 — Depression angle computation for bottom bounce paths

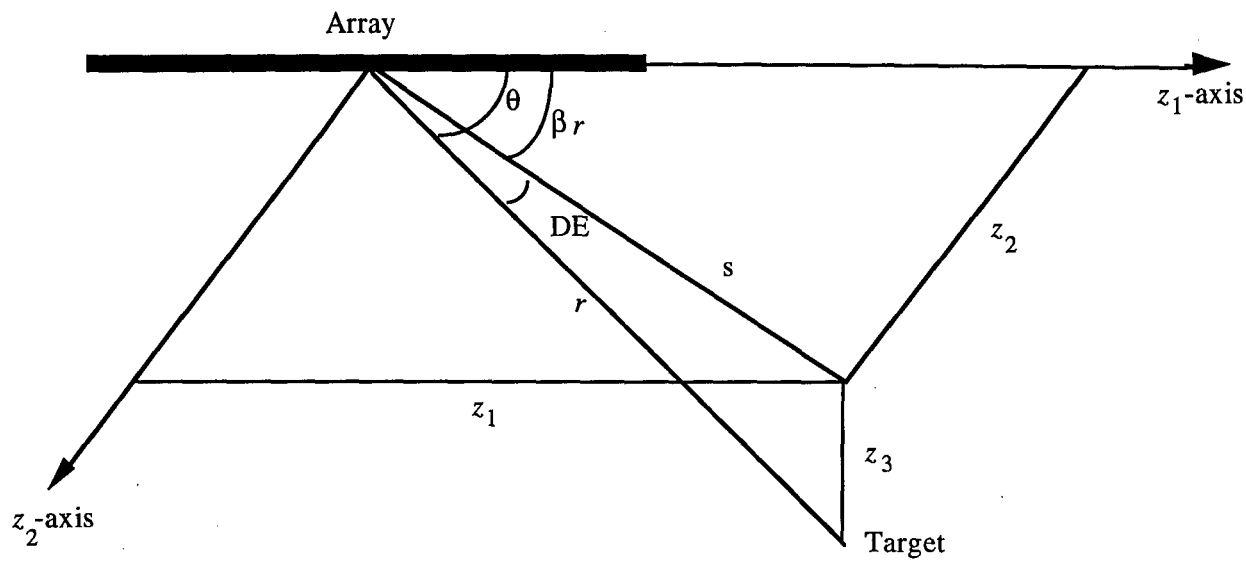
Establish a span of ranges where bottom bounce paths are possible. Suppose this span is $[R_1,R_2]$. For each target state x, where x is a six-dimensional vector (z,v,a) (where $z = (z_1, z_2, z_3)$) in the Nodestar state space, calculate the two-dimensional range $r(x)$ of the state from the towed array. If $r(x)$ is not in $[R_1,R_2]$, then calculate the likelihood as described earlier. If $R_1 \le r(x) \le R_2$, then

compute the apparent depth $d = 2D\text{-}z_r\text{-}z_3$ and the apparent angle $\theta(x)$ from the relationships in Fig. 7, i.e., $\cos^{-1}$

$$\theta(x) = \cos^{-1}\left(\frac{z_1}{\sqrt{z_1^2 + z_2^2 + z_3^2}}\right). \tag{35}$$

One then compares $\theta(x)$ to the observed angle θ in calculating the Nodestar likelihood. If for example, the observation error ε is normal with mean 0 and standard deviation σ, then the likelihood for state x is equal to

$$L(x) = \frac{1}{\sqrt{2\sigma}} \exp\left(\frac{-(\theta(x) - \theta)^2}{2\sigma^2}\right).$$



r = three-dimensional range from array center to

s = two-dimensional range to

θ = three-dimensional bearing to

βr = two-dimensional bearing to

DE = Depression / elevation

$\cos\theta = z_1/r$, $\cos\beta r = z_1/s$, $\cos(DE) = s/r$, $\cos\theta = \cos\beta r \cos(DE)$

Fig. 7 — Three-dimensional angle measured by an array

5.2.2 Bearings to Surface Ships

If one is tracking a surface ship with an array (fixed or mobile) that is at depth D, then one calculates $\theta(x)$ using Eq. (35) with $z_3 = D$. Observe that when z_3^2 is small compared to $z_1^2 + z_2^2$, the three-dimensional bearing is approximately equal to the two-dimensional bearing.

UNCLASSIFIED

5.3 Detection / No Detection Likelihood

This section describes the likelihood function that is implemented in Nodestar to represent the detection/no detection information from a passive acoustic sensor such as a towed or stationary array.

Consider a single (moving or stationary) sensor and let

$$\mathbf{x} = (x_1, ..., x_n)$$

be the vector of target states at times t_i, $i = 1, ..., n$ and

$$\mathbf{D} = (D_1, ..., D_n)$$

be the vector of detection/no detection states at the sensor at t_i, $i = 1, ..., n$. We set $D_i = 1$ or 0 to represent detection or nondetection at time t_i. In this section, we develop a method for computing $Pr\{\mathbf{D}|\mathbf{x}\}$, i.e., the probability of observing a given set of detections and nondetections at the sensor conditioned on the target's path $\boldsymbol{x}$.

5.3.1 Detection Model

The model that we use for detection is the discrete time λ-σ process developed by J. D. Kettelle and first studied extensively in Loane et al. (1964).

For this model, we assume that from the position of the sensor and the target at time t_i, one can compute an instantaneous probability of detection p_i for $i = 1, ..., n$. This is typically done by calculating the mean signal excess $\overline{SE}_i$ in dB from the passive sonar equation, e.g.,

$$\overline{SE}_i = SL - TL - LE - DT,$$

where

SL = source level
TL = transmission loss
LE = $(AN\text{-}DI)\oplus(SN\text{-}AG)$
AN = ambient noise
DI = directivity index
SN = self noise
AG = array gain
DT = detection threshold (or recognition differential)

and $\oplus$ indicates power sum.

In the spotlight version of Nodestar, $\overline{SE}_i$ is computed by using the SPARS performance predictions for the stationary and mobile arrays of the Integrated Underwater Surveillance System (IUSS). SPARS provides the LE and TL terms. Nodestar supplies the DT and SL terms. DT is obtained from an internal table that gives DT (also called recognition differential, RD) as a function of the signal processing resources that are on the beam covering the geographical location of the state x and the designated frequency. If there has been a detection, the designated frequency is the one on which the detection has taken place. If there has been no detection, then the frequency is a narrowband, speed-independent frequency designated for the class of submarine given in the target state. Recall that target class is the sixth dimension of the target state in Nodestar. The source level term SL is obtained from a threat database that provides estimated source levels as a function of frequency for each target class.

We assume that the actual signal excess SE_i is given by

$$SE_i = \overline{SE}_i + \xi_i \tag{36}$$

where $\xi_i \sim N(0, \sigma^2)$. (Note, $N(0, \sigma^2)$ denotes a normal random variable with mean 0 and variance σ^2.) A detection occurs when $SE_i > 0$. Thus

$$p_i = Pr\{SE_i > 0\}. \tag{37}$$

In order to compute the joint probability of a sequence of observations, i.e. $\mathbf{D} = (D_1, ..., D_n)$, we must specify the time dependent correlation in the SE_i's. That is, we must specify the stochastic process that yields the sequence $\{\xi_i, i = 1, ..., n\}$ in Eq. (36). This is done as follows: Let $\{Y_0, Y_1, ...\}$ be a sequence of independent, identically distributed, $N(0, \sigma^2)$ random variables. At each time increment t_i to t_{i+1}, there is the probability

$$\beta = 1 - \exp(-\lambda\Delta) \tag{38}$$

of having a "jump." Let N_i = the number of jumps in $[0, t_i]$. Define

$$\xi_0 = Y_0$$

$$\xi_i = Y_{N_i} \qquad \text{for } i = 1, 2, \ldots .$$

When a "jump" occurs, an independent value of ξ_i is chosen. In this case, we say there has been an acoustic fluctuation. These fluctuations are used to model the random fluctuations in SNR that are observed for underwater acoustics. Typically λ = 1/hr for a moving sensor and σ is in the range of 4 to 9 dB.

With this model, the event $\{D_1 = 1, D_2 = 0, D_3 = 1\}$ is equivalent to the event $\{SE_1 > 0, SE_2 \leq 0, SE_3 > 0\}$. Loane et al. (1964) is concerned with calculating $Pr\{D_i = 0 \text{ for } i = 1, ..., n\}$, i.e., the probability of no detection in the interval $[0, t_n]$. One minus this probability is called cumulative detection probability (cdp). Thus, cdp is the probability of at least one detection in the interval $[0, t_i]$. Appendix A of Curtis et al. (1966) gives an algorithm for calculating cdp for this situation. However, this algorithm is limited to the case where $D_i = 0$ for $i = 1, ..., n$ and is too computationally intensive for use in Nodestar.

5.3.2 Recursive Algorithm

For Nodestar, we will use the recursive algorithm described below. This algorithm is based on the approximation of assuming that D_i is conditionally independent of D_{i-n} and x_{i-n} for $n \geq 2$ given D_{i-1} and x_{i-1}. To calculate $Pr\{\mathbf{D} \mid \mathbf{x}\}$, we calculate the four conditional probabilities given below:

$$\begin{aligned}
b(0, 0, x_i, x_{i-1}) &\equiv Pr\{D_i = 0 \mid D_{i-1} = 0, x_i, x_{i-1}\} \\
b(1, 0, x_i, x_{i-1}) &\equiv Pr\{D_i = 1 \mid D_{i-1} = 0, x_i, x_{i-1}\} \\
b(0, 1, x_i, x_{i-1}) &\equiv Pr\{D_i = 0 \mid D_{i-1} = 1, x_i, x_{i-1}\} \\
b(1, 1, x_i, x_{i-1}) &\equiv Pr\{D_i = 1 \mid D_{i-1} = 1, x_i, x_{i-1}\}.
\end{aligned} \tag{39}$$

UNCLASSIFIED

To calculate $b(0, 0, x_i, x_{i-1})$, we observe that given the position of the target at times t_i and t_{i-1} (i.e., given x_i and x_{i+1}) and knowing the position of the sensor at these times, we can calculate the instantaneous detection probabilities p_i and p_{i-1} at these times. We now break the calculation into two cases.

Case 1. A jump occurred between t_{i-1} and t_i. In this case, detection at t_i is independent of detection at time t_{i-1} so we obtain simply

$$Pr\,\{D_i = 0 \mid D_{i-1} = 0, x_i, x_{i-1} \text{ and a jump}\} = 1 - p_i.$$

Case 2. No jump occurred. In this case, $\xi_i = \xi_{i-1}$, and

$$Pr\,\{D_i = 0 \mid D_{i-1} = 0, x_i, x_{i-1} \text{ and no jump}\}$$

$$= \frac{Pr\,\{\xi_{i-1} \le -\overline{SE}_i \text{ and } \xi_{i-1} \le -\overline{SE}_{i-1}\}}{Pr\,\{\xi_{i-1} \le -\overline{SE}_{i-1}\}}$$

$$= \min\,\{(1 - p_i), (1 - p_{i-1})\} / (1-p_{i-1}).$$

Since β is the probability of a jump occurring in t_{i-1} to t_i, we can combine the two cases to obtain

$$b\left(0,0,x_i,x_{i-1}\right) = \beta\left(1-p_i\right) + \left(1-\beta\right)\frac{\min\left(1-p_i\right),\left(1-p_{i-1}\right)}{1-p_{i-1}}$$

$$= \beta(1-p_i) + (1-\beta)\frac{1-\max\{p_i,p_{i-1}\}}{1-p_{i-1}}. \tag{40}$$

In a similar fashion, we can compute

$$b(1,0,x_i,x_{i-1}) = \beta p_i + (1-\beta)\frac{\max\{p_{i-1},p_{i-1}\} - p_{i-1}}{1-p_{i-1}} \tag{41}$$

$$b(0,1,x_i,x_{i-1}) = \beta(1-p_i) + (1-\beta)\frac{p_{i-1} - \min\{p_i,p_{i-1}\}}{p_{i-1}} \tag{42}$$

$$b(1,1,x_i,x_{i-1}) = \beta p_i + (1-\beta)\frac{\min\{p_i,p_{i-1}\}}{p_{i-1}}. \tag{43}$$

In addition, we define

$$\begin{aligned} b_1(D_1, x_1) = {} & p_1 && \text{if } D_1 = 1, \\ & 1 - p_1 && \text{if } D_1 = 0. \end{aligned} \tag{44}$$

From Eqs. (40) to (44) it is evident that b and b_1 depend on x_i, x_{i-1} and the sensor position at times t_i and t_{i-1}, only through p_i and p_{i-1}. For convenience, we write an alternate form of Eqs. (40) to (44) as a function of D_i , D_{i-1}, p_i , and p_{i-1}. Specifically, let

$$B(0,0,p_i,p_{i-1}) = \beta(1-p_i) + (1-\beta)\frac{1-\max\{p_i,p_{i-1}\}}{1-p_{i-1}}. \tag{40'}$$

$$B(1,0,p_i,p_{i-1}) = \beta p_i + (1-\beta)\frac{\max\{p_i,p_{i-1}\} - p_{i-1}}{1-p_{i-1}} \tag{41'}$$

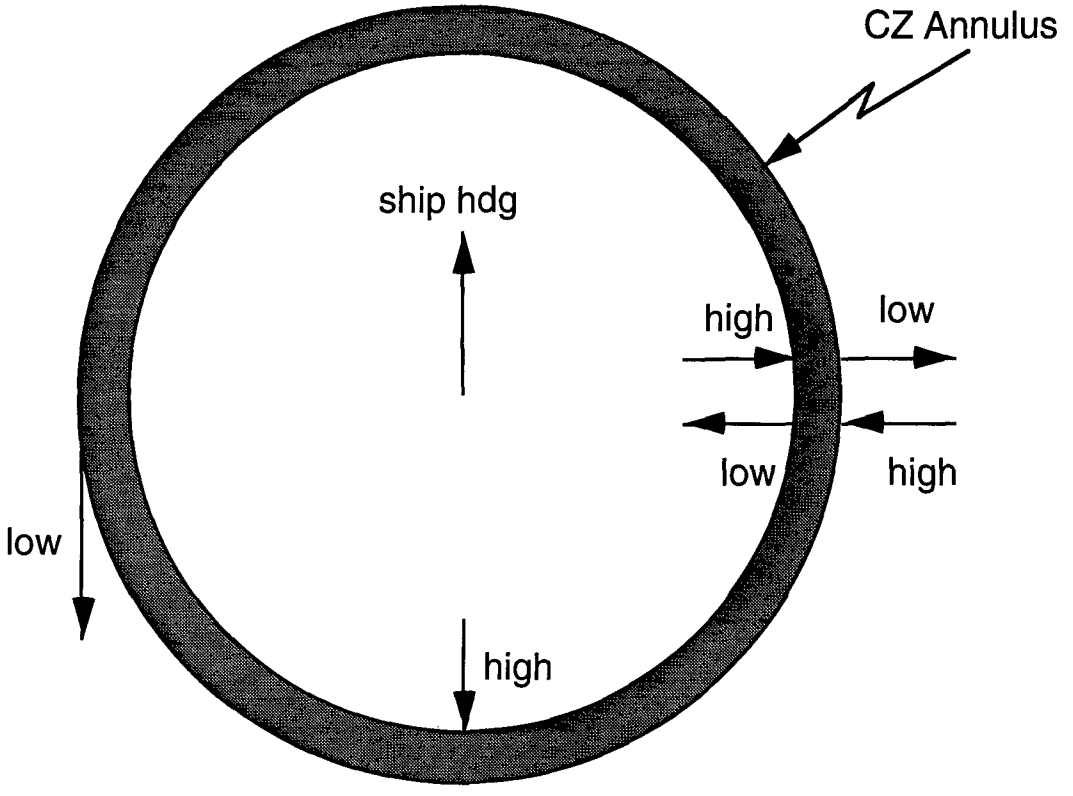
$$B(0,1,p_i,p_{i-1}) = \beta(1-p_i) + (1-\beta)\frac{p_{i-1} - \min\{p_i,p_{i-1}\}}{p_{i-1}} \tag{42'}$$

$$B(1,1,p_i,p_{i-1}) = \beta p_i + (1-\beta)\frac{\min\{p_i,p_{i-1}\}}{p_{i-1}} \tag{43'}$$

$$\begin{aligned} B_1(D_1,p_1) &= p_1 \quad && \text{if } D_1 = 1 \\ 1\text{-}p_1 & && \text{if } D_1 = 0. \end{aligned} \tag{44'}$$

From the above discussion we can see that this likelihood function depends on both position and velocity. As an example, consider the situation where one has just gained contact , e.g., $D_i = 1$, $D_{i-1} = 0$. Suppose that the sensor is a towed array and that there is a convergence zone present as shown in Fig. 8 (shaded area). This likelihood function will be high for position and velocity combinations that have the target just entering the convergence zone (CZ) and low for those just leaving. Consider for example, a target located just outside the CZ annulus at t_{i-1} with velocity pointing toward the sensor as shown in Fig. 8. This position velocity combination will have a high likelihood. By contrast the same position coupled with a velocity pointing away from the CZ will have a low likelihood.

UNCLASSIFIED



Notes: 1. Target velocities are relative to ship velocity
2. High / low indicates relative likelihood values

Fig. 8 — Position-velocity dependence of likelihood function for gain of detection

We can now calculate

$$Pr\,\{\mathbf{D}|\mathbf{x}\} = b_1(D_1, x_1) \prod_{i=2}^{n} b(D_i, D_{i-1}, x_i, x_{i-1}). \tag{45}$$

Let T be the transition function for the Markov process representing target motion. In Nodestar this process has a six-dimensional state space. We use x_i to denote these target states at time t_i. Thus, we may calculate p_i from knowledge of x_i as needed for the above calculation. Let q_1 denote the probability distribution on target state at time 1. Then

$$Pr\,\{\mathbf{D} \text{ and } \mathbf{x}\} = b_1(D_1, x_1)\, q_1\,(x_1) \prod_{i=2}^{n} T\,(x_{i-1}, x_i) \prod_{i=2}^{n} b(D_i, D_{i-1}, x_i, x_{i-1}). \tag{46}$$

The Markovian nature of Eq. (46) allows us to perform updates recursively for the detection/no detection information. Suppose that q_i is the posterior distribution on target state at time t_i and that we observe detection information D_{i+1} at time t_{i+1}. The update proceeds in three steps:

1. *Motion Update.* Calculate

$$q^-_{i+1}(x) = \sum_{y \in X} q_i(y)\, T(y,x) \text{ for } x \in \mathbf{X} \tag{47}$$

where X is the set of all possible target states.

2. *Information Update.*

$$q^+_{i+1}(x) = b(D_{i+1}, D_i, x, x_i)\, q^-_{i+1}(x) \qquad \text{for } x \in X. \tag{48}$$

3. *Renormalize* to obtain q_{i+1}. In Eq. (48) it may appear that we need to know more than the state x at time t_{i+1} to compute b. But, in fact, the state at time t_{i+1} contains the velocity of the target, and the motion model assumes that the target holds a constant velocity during (t_i, t_{i+1}). Thus, we may compute the target position at time t_i from the position and velocity at time t_{i+1}. As a result, b depends only on D_{i+1}, D_i and x at time t_{i+1}.

5.4 Frequency Likelihood

This section describes the frequency likelihood function that has been implemented in the Spotlight version of Nodestar.

5.4.1 Description of Frequency Likelihood

This likelihood function is calculated when the source producing the detected frequency is identified as to whether it is a part of the propulsion system, a generator, or some other mechanical component of the submarine. An acoustic contact in this case consists of both a measured frequency and an identified mechanical source of that measured frequency. The method for computing the frequency likelihood function applies to both speed-dependent and speed-independent sources.

Recall that the target state $x = (z,v,c)$ where

z = (z_1, z_2, z_3)
z_1 = longitude
z_2 = latitude
z_3 = depth
v = two-dimensional velocity of the target
c = class of the target.

In the threat or target database, each target class c has associated to it a set of speed-dependent and speed-independent sources. Speed-dependent source frequencies (and their harmonics) are stored as linear functions of speed. Sound pressure levels for these sources are stored as functions of speed and aspect. Speed-independent source frequencies (and their harmonics) are stored as expected min and max frequencies with sound pressure levels depending on aspect.

Suppose there are a total of N legitimate source types available for each target class. Let n denote the index running over source type.

Let

UNCLASSIFIED

$q_t(x)$ = $Pr\{$Target state x at time $t\}$

$L((n,\gamma),x)$ = $Pr\{$observing source n at frequency γ given target in state $x\}$

$s(x)$ = speed of target in state x

$\alpha(x)$ = aspect of target to sensor in state x.

Suppose we receive a signal at frequency γ at a sensor identified as having been produced by a source of type n. We assume that the measurement error in the frequency is normally distributed with mean 0 and standard deviation σ_f. Let σ_a be the standard deviation of acoustic fluctuations.

For each target state x, we compute the following:

$\gamma_n(x)$ = expected frequency of source type n from target in state x

$\Delta_n(x) = \Delta_n(z,v,c)$ = Doppler shift at the sensor given a target of class c, position z, and velocity v is radiating at frequency $\gamma_n(x)$.

$$L((n,\gamma),x) = \varphi\left(\frac{\gamma - \gamma_n(x)\Delta_n(x)}{\sigma_f}\right), \tag{49}$$

where

φ = density function for a normal (0,1) random variable

and L gives the likelihood of observing frequency γ when the actual frequency is $\gamma_n(x) + \Delta_n(x)$.

Strictly speaking, the term σ_f should represent the measurement error associated with the sensor. Consider the case of a speed-independent source with significant variability in the expected source frequency. One would like to assign a high likelihood to measurements of states that could have produced such data. If the source frequency characteristically falls in a relatively wide band around a particular frequency, it would be desirable to increase the error σ_f of the distribution. Thus, a relatively high likelihood could still be assigned to a measured frequency which differed (by more than just the measurement error) from the value stored in the database.

The easiest way to accomplish this would be to multiply σ_f by some number $\rho > 1$ that depended on the variability of the expected source frequency. The right way to do it would involve placing an a priori distribution on the frequency in the database (with standard deviation representing something about the expected spread in the source frequency) and then computing the standard deviation of the posterior distribution of the source frequency given the measurement with error σ_f. However, this approach would entail other, more complicated modifications to the above formulas and would be of dubious utility.

5.4.2 Implementation of Frequency Likelihood in Nodestar

The frequency likelihood in Eq. (49) is extremely sensitive to target velocity, particularly if σ_f is small, so it is necessary to evaluate this function more carefully than any of the other likelihood functions in Nodestar. We have chosen to maximize the value of this function over the region of velocity space being considered rather than simply evaluating it at the center of the region. This can be done analytically and without too much computational cost.

5.4.3 Description of Likelihood Computation

To maximize the likelihood in a velocity cell covering $\{ r_0 \le r \le r_1 , \theta_0 \le \theta \le \theta_1 \}$, we can find the point(s) at which $f(r, \theta) = [\gamma - \gamma_n(x) - \Delta_n(x)]^2$ is at a minimum.

The expression for passive Doppler shift is

$$\Delta = \gamma_n(x)\left(\frac{1}{1-\beta(\mathbf{v}-\mathbf{v}_s)\bullet\mathbf{B}} - 1\right),$$

where

$\mathbf{B}$ = Unit vector from z (target location) to the sensor

$\mathbf{v}_S$ = Sensor velocity

β = (Speed of sound)$^{-1}$.

Note that all vector operations assume a locally flat Earth. For $(\mathbf{v} - \mathbf{v}_S)\bullet\mathbf{B} << 1/\beta$, we can approximate

$$\Delta = \beta\, \gamma_n(x)\, (\mathbf{v} - \mathbf{v}_S)\bullet\mathbf{B}$$

which is easier to deal with. For typical submarine speeds, the error introduced by this approximation is << 1 percent. We will therefore minimize the function:

$$f(r, \theta) = (\gamma - \gamma_n(x) - \beta\, \gamma_n(x)\, (\mathbf{v} - \mathbf{v}_S)\bullet\mathbf{B})^2 .$$

5.4.4 Speed-Independent Sources

For speed-independent sources, γ_n is independent of target state.

$$f(r, \theta) = [\gamma - \gamma_n + \beta\, \gamma_n\, \mathbf{v}_S \bullet\mathbf{B} - \beta\, \gamma_n\, r \cos(\theta - B_\theta)]^2$$

$$= [A + B\, r \cos(\theta')]^2,$$

where

A = $(\gamma - \gamma_n + \beta\, \gamma_n\, \mathbf{v}_S \bullet\mathbf{B})$

B = $-\beta\, \gamma_n$

B_θ = direction of $\mathbf{B}$

θ' = $\theta - B_\theta$.

This function's only extrema are the points for which $r \cos(\theta') = -A/B$. Note that for this set of points $f(r, \theta) = 0$, so they specify the global minimum. There are no local minima.

UNCLASSIFIED

The shape of this curve is such that it is impossible to draw a finite region in (r, θ) space whose interior contains points on the curve but does not intersect the curve at least twice. Therefore, no velocity cell can contain a minimum without that minimum also being on the perimeter of the cell. We may compute the minimum over the cell by computing the minimum on the perimeter.

To find the constrained minimum along the perimeter of the cell, we have to check the following points:

A) The cell's corners:

$$\{ f(r_0, \theta_0), f(r_0, \theta_1), f(r_1, \theta_0), f(r_1, \theta_1) \}$$

B) Constrained minimum of $f(r_0, \theta)$ in $\{ \theta_0 < \theta < \theta_1 \}$:

i) $f(r_0, B_\theta)$, if $\{ \theta_0 < B_\theta < \theta_1 \}$ and $\{ A > -Br_0 \}$

ii) $f(r_0, \mathbf{B}_\theta + \pi)$, if $\{ \theta_0 < B_\theta + \pi < \theta_1 \}$ and $\{ A < Br_0 \}$

* iii) $f(r_0, \cos^{-1}(-\frac{A}{Br_0}))$,

if $\{ -1 \le \frac{A}{Br_0} \le 1 \}$ and $\{ \theta_0 < \cos^{-1}(-\frac{A}{Br_0}) < \theta_1 \}$.

C) Constrained minimum of $f(r_1, \theta)$ in $\{ \theta_0 < \theta < \theta_1 \}$:

i) $f(r_1, B_\theta)$, if $\{ \theta_0 < B_\theta < \theta_1 \}$ and $\{ A > -Br_1 \}$

ii) $f(r_1, \mathbf{B}_\theta + \pi)$, if $\{ \theta_0 < B_\theta + \pi < \theta_1 \}$ and $\{A < Br_1 \}$

* iii) $f(r_1, \cos^{-1}(-\frac{A}{Br_1}))$,

if $\{ -1 \le -\frac{A}{Br_1} \le 1 \}$ and $\{ \theta_0 < \cos^{-1}(-\frac{A}{Br_1}) < \theta_1 \}$

D) Constrained minimum of $f(r, \theta_0)$ in $\{ r_0 < r < r_1 \}$:

* $f(\frac{A}{B\cos(\theta_0 - B_\theta)}, \theta_0)$, if $\{r_0 < \frac{A}{B\cos(\theta_0 - B_\theta)} < r_1\}$

E) Constrained minimum of $f(r, \theta_1)$ in $\{ r_0 < r < r_1 \}$:

* $f - \left(\frac{A}{B\cos\left(\theta_1 - B_\theta\right)}, \theta_1 \right)$ if $\left\{ r_0 < -\frac{A}{B\cos\left(\theta_1 - B_\theta\right)} < r_1 \right\}$

* Note that f evaluates to the global minimum in this case, so we can stop computation here if the above condition is met.

5.4.5 Speed-Dependent Sources

For speed-dependent sources, γ_n is proportional to target speed. We make the substitution

$$\gamma_n = \alpha\, r,$$

where α is the Hz per knot of the assumed source type. Then

$$\begin{aligned} f(r, \theta) &= [\gamma - \gamma_n(\mathbf{x}) + \beta\, \gamma_n(\mathbf{x})\, \mathbf{v}_S \bullet \mathbf{B} - \beta\, \gamma_n(\mathbf{x})\, r \cos(\theta - \mathbf{B}_\theta)]^2 \\ &= [A + Br + Cr^2 \cos(\theta')]^2 , \end{aligned}$$

where

$$\begin{aligned} A &= \gamma \\ B &= \alpha\,(\beta\, v_S \bullet \mathbf{B} - 1) \\ C &= -\beta\alpha \\ \mathbf{B}_\theta &= \text{Direction of } \mathbf{B} \\ \theta' &= \theta - \mathrm{B}_\theta\ . \end{aligned}$$

As in the speed-independent case, this function's only extrema are the set of points for which $f(r, \theta) = 0$, or when $A + Br + Cr^2 \cos(\theta') = 0$. And, although this curve is more difficult to visualize, it is also infinite, and its only discontinuities tend toward $r = \infty$. Using the same argument as the previous case, we need only find the constrained minimum on the cell's perimeter. The points we need to consider are:

A) The cell's corners:

$$\{ f(r_0, \theta_0), f(r_0, \theta_1), f(r_1, \theta_0), f(r_1, \theta_1) \}$$

B) Constrained minimum of $f(r_0, \theta)$ in $\{ \theta_0 < \theta < \theta_1 \}$:

i) $f(r_0, \mathbf{B}_\theta)$, if $\{ \theta_0 < \mathbf{B}_\theta < \theta_1 \}$ and $\{ A + Br_0 > -1 \}$

ii) $f(r_0, \mathbf{B}_\theta + \pi)$, if $\{ \theta_0 < \mathbf{B}_\theta + \pi < \theta_1 \}$ and $\{ A + Br_0 < 1 \}$

* iii) $f\left(r_0, \cos^{-1}\left(\frac{-(A + Br_0)}{Cr_0^{\,2}}\right)\right)$,

$$\text{if} \left\{ -1 \le \frac{\left(A + B_{r_0}\right)}{C_{r_0}^{\ 2}} \le 1 \right\} \text{ and } \left\{ \theta_0 \le \cos^{-1}\left(\frac{-A + B_{r_0}}{C_{r_0}^{\ 2}} \right) \le \theta_1 \right\}$$

C) Constrained minimum of $f(r_1, \theta)$ in $\{\theta_0 < \theta < \theta_1\}$:

UNCLASSIFIED

i) $f(r_1, \mathbf{B}_\theta)$, if $\{ \theta_0 < \mathbf{B}_\theta < \theta_1 \}$ and $\{ A + Br_1 > -1 \}$*

ii) $f(r_1, \mathbf{B}_\theta + \pi)$, if $\{ \theta_0 < \mathbf{B}_\theta + \pi < \theta_1 \}$ and $\{ A + Br_1 < 1 \}$

* iii) $f(r_1, \cos^{-1}\left(\frac{(A+Br_1)}{Cr_1^2}\right))$,

if $\{ -1 \le \frac{(A + Br_1)}{Cr_1^2} \le 1 \}$ and $\{\theta_0 \le \cos^{-1}\left(\frac{(A+Br_0)}{Cr_1^2}\right) \le \theta_1\}$

D) Constrained minimum of $f(r, \theta_0)$ in $\{ r_0 < r < r_1 \}$:

* i) $f\left(r_{\text{trial}} = \frac{\sqrt{B^2 - 4AC\cos(\theta_0 - \mathbf{B}_\theta)} - B}{2C\cos(\theta_0 - \mathbf{B}_\theta)}, \theta_0\right)$, if $\left\{r_0 < r_{\text{trial}} < r_1\right\}$

* ii) $f\left(r_{\text{trial}} = \frac{-\sqrt{B^2 - 4AC\cos(\theta_0 - \mathbf{B}_\theta)} - B}{2C\cos(\theta_0 - \mathbf{B}_\theta)} \theta_0\right)$, if $\left\{r_0 < r_{\text{trial}} < r_1\right\}$

iii) $f\left(-\frac{B}{2C\cos(\theta_0 - \mathbf{B}_\theta)}, \theta_0\right)$, if $\left\{r_0 < -\frac{B}{2C\cos(\theta_0 - \mathbf{B}_\theta)} < r_1\right\}$

E) Constrained minimum of $f(r, \theta_1)$ in $\{ r_0 < r < r_1 \}$:

* i) $f\left(r_{\text{trial}} = \frac{\sqrt{B^2 - 4AC\cos(\theta_1 - \mathbf{B}_\theta)} - B}{2C\cos(\theta_1 - \mathbf{B}_\theta)}, \theta_1\right)$, if $\left\{r_0 < r_{\text{trial}} < r_1\right\}$

* ii) $f\left(r_{\text{trial}} = \frac{\sqrt{B^2 - 4AC\cos(\theta_1 - \mathbf{B}_\theta)} - B}{2C\cos(\theta_1 - \mathbf{B}_\theta)}, \theta_1\right)$, if $\left\{r_0 < r_{\text{trial}} < r_1\right\}$

iii) $f\left(-\frac{B}{2C\cos(\theta_1 - \mathbf{B}_\theta)}, \theta_0\right)$, if $\left\{r_0 < -\frac{B}{2C\cos(\theta_1 - \mathbf{B}_\theta)} < r_1\right\}$

Once we have found the minimum value of f in the cell, we convert it to a likelihood:

* Note that f evaluates to the global minimum in this case, so we can stop computation here if the above condition is met.

$$L = \frac{1}{\sqrt{2\pi}\sigma_f} exp\left(\frac{-f_{min}^2}{2\sigma_f^2}\right).$$

5.5 Land Avoidance Likelihood

Nodestar uses two forms of the land avoidance likelihood function. The first form depends only on position. The second depends on position and velocity.

5.5.1 Position Only Land Avoidance Likelihood

Let L_a denote the land likelihood function. The land likelihood function that depends only on target position operates in the following manner. Let

x	= target state
	= (z,v,a) where z is position, v is velocity, and a is attribute
$D(x)$	= distance of x from nearest point of land
d_s	= "shallow" distance
d_d	= "deep" distance where $d_d > d_s$
α	= avoidance function. This is a function such that

$$\alpha(r) = \begin{cases} 0 & \text{for } 0 \le r \le d_s \\ 1 & \text{for } d_d \le d < \infty, \end{cases}$$

and

$\alpha(r)$ increases monotonically from 0 to 1 as r increases from d_s to d_d.

Then

$$L_a(x) = \alpha(D(x)) \text{ for } x \in \boldsymbol{X}. \tag{50}$$

5.5.2 Position and Velocity Dependent Land Avoidance Likelihood

This form of the land avoidance likelihood depends on the position and the velocity of the target state. The velocity dependence also involves the target motion model. In particular, it depends on $\tau(x) = 1/\lambda(x)$, the mean time between velocity changes given the target is in state x. In this case,

$$L_a(x) = \alpha(D(x))\alpha\big(D(x + \tau(x)v)\big) \text{ for } x \in \boldsymbol{X}. \tag{51}$$

The intuitive meaning of Eq. (50) is that we multiply the position-only likelihood by a factor that is equal to the avoidance function evaluated at the position obtained by projecting the state ahead at its velocity v for a time equal to the mean time between velocity changes for that state. If this projected position is close to land, then this state (i.e., combination of position and velocity) receives a low likelihood value. Thus, position-velocity combinations that tend to put the target near land become unlikely.

5.6 Elliptical Contact Likelihood

Nodestar can fuse the standard contact information that is provided by many sensors in the form of a position and an elliptical uncertainty region. This is done by converting the elliptical contact

UNCLASSIFIED

information into a bivariate normal distribution as follows. The reported position (lon_0, lat_0) is the mean of the distribution. The uncertainty region is defined by an ellipse that is specified by the length in nautical miles of the semi-major and semi-minor axes and the orientation of the major axis measured clockwise from north. This ellipse is interpreted as the two-standard deviation ellipse of a bivariate normal distribution centered at (lon_0, lat_0). (Note, since ellipses tend to be small in extent, we use a local flat Earth approximation in the plane tangent to (lon_0, lat_0) for constructing the bivariate normal distribution.) The ellipse is converted to the covariance matrix Σ of the bivariate normal distribution in the local flat Earth coordinates with origin at (lon_0, lat_0) and axes parallel to the north-south and east-west directions. Let $h(\cdot, \Sigma)$ be the density function for the bivariate normal distribution with mean $(0,0)^T$ and covariance matrix Σ. Let

$lat(x)$ = latitude of the state x

$lon(x)$ = longitude of the state x.

Then the likelihood function L for this contact is

$$L(x) = h(\Delta(x), \Sigma) \text{ for } x \in \boldsymbol{X}, \quad \text{where} \quad \Delta(x) = \begin{pmatrix} 60\cos(lat_0)(lon(x) - lon_0) \\ lat(x) - lat_0 \end{pmatrix}. \tag{52}$$

5.7 ELINT Likelihood

Nodestar has an ELINT likelihood function that uses the information about the type of radar that was detected as well as the locating information in the detection. Usually the locating information is in the form of an elliptical contact. For this piece of information we use the elliptical contact likelihood given above. If, in addition, the type of radar detected is identified, we multiply the contact likelihood by the radar type likelihood function. To use this function, we must have a database that lists, for each radar type, all the classes of submarines (or surface ships) that carry that type of radar. The radar type likelihood function l is defined on the attribute component a of the state x. Recall that a is the target class. Suppose the radar is identified as type k. Then

$$l(k, x) = \begin{cases} 1 & \text{if class } a \text{ has a radar of type } k \\ 0 & \text{if class } a \text{ does not have a radar of type } k \end{cases} \quad \text{for } x \in \boldsymbol{X}, \tag{53}$$

where $x = (z, v, a)$. The effect of applying the likelihood function l is to set to zero the probability on any state whose class a does not carry radar type k.

6. MULTIPLE TARGET VERSION OF NODESTAR

The Nodestar engine provides for multiple target correlation and tracking. In this section, we describe the data association algorithms used and the method Nodestar uses for computing multiple target state distributions.

Recall that the process of data association is equivalent to partitioning the set of observations into disjoint sets of associated contacts. For each set, Nodestar creates a track object. This is a C^{++} object, i.e., an object in the computer science sense. Each track object is an independent copy of a Nodestar single-target tracker. It has all the properties described in Sections 4 and 5. Multiple-target Nodestar is designed so that these tracks can be computed on a single CPU or distributed across multiple ones.

With this said, the main question that remains to be answered about multiple-target Nodestar is: how does it perform its data association?

6.1 Nonlinear Data Association

Nodestar separates the process of data association and state estimation in the manner discussed in Section 2. It is a single-hypothesis, correlator-tracker in the lexicon of that section. For each contact, the probabilities of association are calculated in a Bayesian fashion and the most likely association hypothesis is chosen. To calculate the association probabilities, Nodestar uses a nonlinear Bayesian approach very similar to the one described in Mori et al. (1986). This is in contrast to the data association computations that are performed for most Kalman-based trackers. Those usually involve calculating a geometric missed distance between two multivariate normal distributions and applying a chi-squared test. This approach is not Bayesian and is only appropriate for linear Gaussian trackers.

6.1.1 Probability Model

Suppose that we obtain an observation at time t and that there are N tracks that exist in Nodestar at the time of the observation. Mathematically each track is a collection of observations that have been associated to a single target. Nodestar uses these observations to produce an estimate of the target's state.

When a contact is obtained, we assume a prior probability distribution $\{r_n : n = 1,\ldots, N+1\}$ for the target that produced the contact. Specifically, we set

r_i = probability that contact was generated by the ith target for $n = 1,\ldots, N$

r_{N+1} = probability contact was generated by a new target.

Note that Nodestar does not distinguish between new targets and false targets. Typically, Nodestar operates with

$$r_{N+1} = \alpha \text{ (for some } 0 < \alpha < 1\text{), and}$$

$$r_n = (1-\alpha)/N \text{ for } n = 1,\ldots, N. \tag{54}$$

In effect, Nodestar treats the new target probability α as a parameter that can be set by the user. We could derive α from a model that estimates the arrival rate of new targets and the generation of contacts from these. Of course, this model would require assumptions and inputs. Since we cannot predict the situations in which Nodestar will be used as a tracker, we prefer to specify α as a direct input. Once α has been specified, it seems reasonable to set the prior probability of the contact being generated by the target corresponding to an existing track equal to $(1-\alpha)/N$ for each track.

6.1.2 Likelihood of a Response

We now calculate the likelihood that a given target (track) produced the contact. As an example, consider a line of bearing detection by a Sound Surveillance Underwater System (SOSUS) array. This contact has two components: the gain of detection and the estimated bearing of the target. These two components comprise the response. Nodestar produces a joint likelihood function for this two-component response.

6.1.2.1 Existing Tracks (Targets)

Let Y = the response obtained at time t, and

UNCLASSIFIED

$L(Y \mid x, n)$ = Likelihood of obtaining response Y given that it was produced by target n in state x.

Let

$q_n^-(t,x)$ = Probability of target n in being in state x at time t.

Then,

$$g(n) = \int_X L(Y \mid x, n) q_n^-(t,x) dx \tag{55}$$

is the likelihood that the response Y is produced by target n, for $n = 1,\ldots,N$.

6.1.2.2 New Targets

We now calculate the likelihood that the contact was generated by a new target. To do this, we need to establish some simple probability models for the generation of contacts by new targets.

6.1.2.2.1 Line of Bearing Detections — Suppose the response is a line of bearing detection, i.e., a gain of contact and an estimated bearing. Let the estimated bearing be θ_0 with measurement error that is normal with mean 0 and standard deviation σ degrees. Suppose that the distribution of new targets is uniform over a disk of radius R so that

$$q_{N+1}^-(t,(r,\theta)) = \frac{r}{180R^2} \quad \text{for } 0 \le \theta \le 360 \text{ and } 0 \le r << R.$$

Suppose that

$$\begin{aligned} L((\theta_0,\text{detection})|(r,\theta)) &= Pr\{\text{response} = (\theta_0,\text{detection}) \mid \text{new target at } (r,\theta)\} \\ &= \tfrac{P_d}{\sigma\sqrt{2\pi}} \exp(\tfrac{-(\theta-\theta_0)^2}{2\sigma^2}) \text{ for } 0 \le \theta \le 360. \end{aligned}$$

Then

$$\begin{aligned} g(N+1) &= \int_0^{360}\int_0^R L((\theta,\text{detection})|(r,\theta)) q_{N+1}^-(t,(r,\theta)) dr d\theta \\ &= \int_0^{360}\int_0^R \tfrac{P_d}{180R^2} \tfrac{1}{\sigma\sqrt{2\pi}} \exp(\tfrac{-(\theta-\theta_0)^2}{2\sigma^2}) r dr d\theta \\ &\approx \frac{P_d}{360} \text{ for } \sigma << \pi. \end{aligned} \tag{56}$$

6.1.2.2.2 Elliptical Contacts — If the reported contact is an elliptical position estimate and the sensor detection probabilities are not modeled in Nodestar, then we compute $g(N+1)$ as follows. We interpret the elliptical contact as corresponding to a position estimate with a bivariate normal error distribution with mean 0 and covariance matrix Σ, where Σ is derived from the ellipse. Let f denote the density function for this distribution. Suppose that the contact is in the form of an ellipse with center point y and associated covariance matrix Σ. Then

$$L((y,\Sigma)|x) = Pr\{\text{elliptical contact}(y,\Sigma)|\ \text{new target at } x\}$$
$$= f(y-x,\Sigma) \text{ for } x \in R^2.$$

We shall assume that the location of a new target is uniform over a region W of area A so that

$$q_n^-(t,x) = \frac{1}{A} \text{ for } x \in W.$$

Then

$$g(N+1) = \int_W \frac{1}{A} f(y-x,\Sigma)dx$$
$$\approx \frac{1}{A} \text{ for } \Sigma \text{ "much smaller" than } A. \tag{57}$$

6.1.2.2.3 Association Probabilities — Having calculated the likelihood of the response Y given it was produced by target n, for $n = 1,\ldots, N+1$, we calculate the posterior probabilities $\tilde{r}_n$ of the contact having been generated by the nth target as follows:

$$\tilde{r}_n = \frac{g(n)r_n}{\sum_{m=1}^{N+1} g(m)r_m} \text{ for } n = 1,\ldots, N+1. \tag{58}$$

Nodestar associates the response to the target having the highest value of $\tilde{r}_n$. In the Spotlight version of Nodestar, users can change or perform associations manually if they desire.

6.1.2.2.4 Creating and Deleting Tracks — If the highest value of $\tilde{r}_n$ occurs for $n = N+1$, then a new track (object) is created. Nodestar also contains logic for deleting tracks. The user can set a time threshold H. If no response has been associated with a track for the last H hours, then the track will be automatically deleted. This logic can be extended to other criteria. For example, Nodestar could be set up so that if the area of the 86 percent containment region (i.e., the smallest set of spatial cells containing 86 percent of the track probability) for a track becomes larger than a designated value, the track is deleted. The user can also delete tracks manually.

6.2 Probabilistic Data Association

The Nodestar methodology allows for a nonlinear version of probabilistic data association (PDA) that is more accurate and natural than the one given by Bar-Shalom and Fortman (1988). The Bar-Shalom version of PDA is applied to Gaussian distributions and produces Gaussian distributions. As a result, it can not reflect the multimodal distributions that should be a natural result of PDA. Note that the methodology described below has not yet been implemented in Nodestar.

Suppose that we have a sensor response that can be associated with two tracks m and n. By this, we mean that sum $\tilde{r}_m + \tilde{r}_n$ is "close" to 1, i.e., the association probabilities for all other tracks are negligible. Let L_n be the likelihood function given the response is associated to target n and L_m the likelihood function given the response is associated to target m. For convenience, let us assume that $\tilde{r}_m$ and $\tilde{r}_n$ have been normalized so that they sum to 1. Define modified likelihood functions $\overline{L}_n$ and $\overline{L}_m$ as follows.

UNCLASSIFIED

$$\bar{L}_n(x) = 1 - \tilde{r}_n + \tilde{r}_n \frac{L_n(x)}{\int_X L_n(x')q_n^-(t,x')dx'}$$
$$\bar{L}_m(x) = 1 - \tilde{r}_m + \tilde{r}_m \frac{L_m(x)}{\int_X L_m(x')q_m^-(t,x')dx'} \quad \text{for } x \in X. \tag{59}$$

We apply $\bar{L}_n$ and $\bar{L}_m$ to tracks n and m, respectively. The result of applying the likelihood function $\bar{L}_n$ in Eq. (59) to the distribution $q_n^-(t,\cdot)$ is a posterior distribution on track n that is a mixture of the posterior distribution that would have been produced if the response had been assigned to track n and the one that would have been produced if the response had not been assigned to n. A similar result holds for the posterior on track m. Note the above process is easily extended to distributing the association over three or more tracks.

7. EXTENSIONS OF NODESTAR

The Nodestar engine and most of the likelihood functions described above have been developed for the Spotlight version of Nodestar. In this chapter we discuss some extensions that have been or can be made to this version of Nodestar. These extensions provide examples of the flexibility and extensibility of Nodestar. They are by no means the only ways that Nodestar can be extended.

The general process of extending Nodestar to include a new type of information is quite simple to outline. For any observation or response representing a new type of information or sensor, one must construct a likelihood function that computes the probability of receiving that response as a function of target state. This usually requires two things: first, a good physical model of the sensor or response process and second, the target state space must contain the right information for the calculation, e.g., velocity, if one wishes to account for Doppler in a frequency reading. If the state space contains the necessary information, then one has only to add a new module for calculating the likelihood function. Having done this, the process of association proceeds as described in Section 6. Once the data is associated to a track, the information update proceeds by using Eq. (13).

This process is very significant. Once the likelihood function has been calculated, the data fusion process (association and state estimation) is performed automatically by the Nodestar engine. This makes the Nodestar engine extremely powerful. It effectively reduces the problem of fusing the information from a new type of sensor to a question of modeling the physics of the sensor, a process that is well understood. Use of this procedure removes the mystery and the need for ad hoc procedures from data fusion. It provides a general approach and framework for performing data fusion.

7.1 Surface Ship Tracking

The problem of surface ship tracking readily lends itself to the Nodestar approach. There are many types of sensors whose information must be fused to obtain tracks and identities for surface ships. Each of these types of information can be fused by using the likelihood approach described in Section 6. We list a number of typical sensors and discuss how information from them can be fused by Nodestar.

State Space. A reasonable state space for surface ship tracking is five-dimensional, i.e., position (2D), velocity (2D), ship class (or name).

Radar. A typical radar report is a position report with an elliptical uncertainty. This is a special case of the elliptical contact likelihood already described.

Acoustic. Acoustic detections may come from fixed or mobile arrays of passive sensors. This information can be modeled using the bearing likelihood functions. If there is a ship database that provides frequency and source level information for the acoustic signals generated by the ships, the frequency likelihood and the detection/no detection likelihood functions can be used.

ELINT. If information is available by ship class or even name, then the ELINT likelihood function described in Section 5.7 can be extended to include not only radar type information but also pulse repetition interval and scan frequency.

Visual. Visual detections provide not only locating information but also information about the class and even name of the ship detected. This information is incorporated into the distribution on ship class or name represented by the fifth dimension in the state space. If the information is good enough, this distribution will have probability one on a single class or ship name for the target detected.

Land Avoidance. Land avoidance is also important for surface ships. The land avoidance likelihood function applies to this problem also.

7.2 Shallow Water Active Acoustic Tracking

In this section, we describe an extension of Nodestar that has been made to tackle the problem of multistatic active tracking of submarines in shallow water. This work has been performed as part of a Phase I Small Business Innovative Research (SBIR) project from Naval Sea Systems Command (NAVSEA) and continues under Phase II funding. This version of Nodestar is called SWATS (Shallow Water Active Tracking System).

A typical scenario for the use of the SWATS tracker might be as follows. We are concerned about tracking diesel submarines in shallow water near land. A radar contact on the submarine's periscope is obtained but lost as the submarine dives. Three or more vertical arrays of hydrophones are dropped in the vicinity of the contact along with an active source pinging at 200 to 300 Hz. We wish to process the acoustic returns at the arrays to track and localize the submarine.

The shallow water environment is a complex and frustrating one for acoustic detection. Understanding shallow water detection requires an understanding of acoustic propagation in shallow water environments. This section presents a brief discussion of normal mode acoustics and describes how it is used in the SWATS tracker to develop a likelihood function appropriate for this problem. A key feature of this likelihood function is that it uses the acoustic time series received at the hydrophones as its input. This provides an example where the detection and tracking function are combined in the tracker.

This section is split into two subsections. The first provides background information about normal mode propagation theory and how normal modes are computed. Additional information about this theory can be obtained from Jensen et al. (1994) and Tolstoy (1993). The second describes how normal modes are used to develop the likelihood function for this situation.

7.2.1 Normal Mode Propagation Theory

For shallow water, acoustic propagation is very complex. As shown in Fig. 9, at ranges beyond several depth-lengths, ray propagation theories quickly become overwhelmed by the multiple surface reflections and bottom bounces. With increased range, ray theory quickly becomes inaccurate.

UNCLASSIFIED

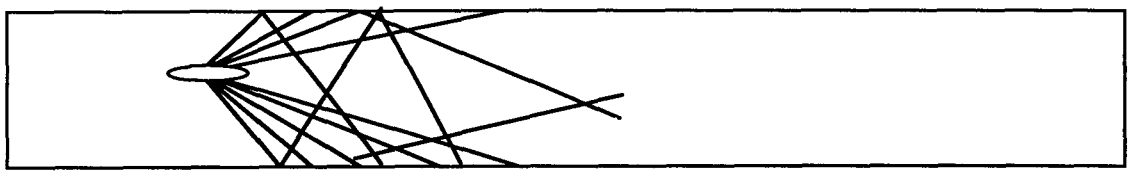


Fig. 9 — Ray propagation of sound from a sound source in shallow water

The solution is to model the entire water column as a shallow waveguide, and to find the normal modes that satisfy the environmental conditions. Each of the normal modes travels in the waveguide at differing speeds from the source and is recombined at the sensor to get the total pressure. Typically, higher order normal modes attenuate quickly with distance so that with increased range, the first few normal modes suffice to describe the sound propagation. This is shown in Fig. 10.

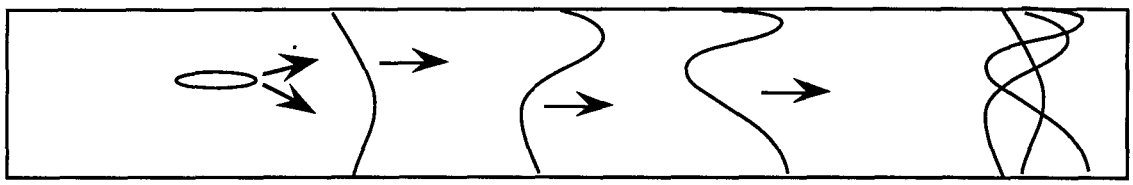


Fig. 10 — Propagation of the first three normal modes from a target

Normal modes are calculated numerically for a given environment and sound frequency. Nodestar uses the KRAKEN normal mode program (Porter (1991) and Porter and Reiss (1984)) which is used by the NRL Acoustics Branch and has been successfully tested against experimental data.

The normal mode program KRAKEN uses the sound speed profile, the attenuation, and the density of both the ocean water and bottom to numerically calculate the modes. The result is a numerical evaluation of each mode $U_m(z)$ over the entire depth range z, and calculation of the complex modal wavenumber k_m.

Using the modes calculated by KRAKEN, we compute the sound pressure ϕ resulting from a source of strength Q_S (measured in pressure at a reference distance, e.g., 1 yd) at frequency ω and depth z_s by

$$\phi(r,z,t) = -i\pi Q_S \sum_m U_m(z_s)\, U_m(z)\, e^{+i\omega t}\, H_o^2(k_m r) \tag{60}$$

$$= Q_S \sum_m \sqrt{\frac{2\pi}{k_m r}}\, U_m(z_s)\, U_m(z)\, e^{+i\omega t}\, e^{-ik_m r}\ ;\ (k_m r) >> 1$$

where t is time, r is the range from the source, and z is the depth at which the pressure is measured. H_0^2 is the Hankel function of the second kind which is approximated for the far-field in the second line. The time dependence is usually dropped in this equation. Note, in the far field limit, that

$$H_0^1(kr) \rightarrow \frac{2}{\pi kr} e^{i(kr-\pi/4)} \text{ and } H_0^2(kr) \rightarrow \frac{2}{\pi kr} e^{i(kr-\pi/4)} \text{ for } kr >> 1.$$

In the case of shallow water, k is usually fairly large. For $k = 0.5m^{-1}$, the condition $kr >> 1$ implies that $r >>$ 2m or that the sensor is much farther than 2 meters from the source.

For a vertical array of n hydrophones placed at successive depths z_h, for $h = 1,\ldots, n$, we have that the pressure reading at hydrophone h is

$$P_h = \sum_m \sqrt{\frac{2\pi}{k_m r_h}}\; Q_s\, U_m(z_s)\, U_m(z_h) \exp(-ik_m r_h) \tag{61}$$

where h indexes the hydrophone at depth z_h and distance r_h from the point source. The vector **P** is used to represent the complex vector which has complex components $P_1, P_2,\ldots, P_n$. We have dropped the time dependence here.

The U_m are numerically computed normal modes for the ocean environment of interest, and are calculated using a normal mode program such as KRAKEN. The normal modes are precomputed once for each environment. Each mode has an associated complex wavenumber k_m that is also computed in the normal mode program. There are typically 30 modes of interest for a problem; higher order modes get attenuated so quickly that they can be neglected. The mode solutions U_m are usually constant or very weak functions of position.

For a *vertical array,* the hydrophone distances r_h are the same for each hydrophone, and the only hydrophone-dependent part of Eq. (61) is the hydrophone depth z_h. Setting $r_h = r$ for each hydrophone, we can rewrite Eq. (61) as

$$P_h = \sum_m A_m\, U_m(z_h)\ , \quad \text{where } A_m \equiv \sqrt{\frac{2\pi}{k_m r}}\; Q_s\, U_m(z_s) \exp(-ik_m r). \tag{62}$$

When sampling the acoustic field with a vertical array of hydrophones, it is convenient to represent the transformation in terms of the *mode amplitudes* $\mathbf{A} = (A_1, A_2, \ldots)$ at the sensor vs the *pressure field* **P** at the sensor. In that case, we can write Eq. (62) as a matrix equation: the pressure vector components $(P_1, P_2,\ldots, P_n)$ are samples of the acoustic *field* from the signal, while the $(A_1, A_2, \ldots)$ are referred to as the acoustic *mode* coefficients of the signal at the sensor. In vector form, Eq. (62) becomes

$$\mathbf{P} = \mathbf{E}\,\mathbf{A} \tag{63}$$

where $\mathbf{E} = (E_{hm})$ and $E_{hm} = U_m(z_h)$. The normal mode propagation model generates a set of modes for the signal source. The *modes* **A** propagate from the source according to the propagation equation described in Eq. (62). When the modes are sampled at the sensor a distance r away from the source, we get the pressure *field* **P**. Most detection methods look at detection in *field* space or in *mode* space.

7.2.2 SWATS Likelihood Function

This subsection describes the likelihood function for SWATS. We first describe the statistical model used to derive the likelihood function and the method for dealing with the ambiguity in target signal strength. Then we present the SWATS likelihood function.

7.2.2.1 Statistical Model of Measurement Error

We assume that the pressure readings are subject to a multivariate normal distribution of error and that we can estimate the parameters of this distribution. The measurement vector **P** is composed of complex pressure readings from each hydrophone in the array. The measured pressure is the sum of signal plus noise. We define the covariance matrix Σ for **P** as

UNCLASSIFIED

$$\Sigma \equiv E\left[\mathbf{P}\mathbf{P}^H\right] - E\left[\mathbf{P}\right]E\left[\mathbf{P}^H\right], \tag{64}$$

where $\mathbf{P}^H$ is used to represent the Hermitian transpose of $\mathbf{P}$, or the complex conjugate of the transpose of $\mathbf{P}$.

Using this, we write the multivariate probability distribution for the measurement $\mathbf{P}$ given a signal $\mathbf{P}_0$ (see Theorem 2.3.1 of Anderson (1984)):

$$Pr\{\mathbf{P}\,|\,\mathbf{P}_0\} = N(\mathbf{P}\,|\,\mathbf{P}_0,\Sigma) = \frac{1}{(\sqrt{2\pi})^n\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(\mathbf{P}-\mathbf{P}_0)^H\Sigma^{-1}(\mathbf{P}-\mathbf{P}_0)\right). \tag{65}$$

7.2.2.2 Signal Strength Ambiguity

In order to use Eq. (65) as a likelihood function, we must be able to predict $\mathbf{P}_0(x)$, the signal at the array given that the target state is x. To do this, we have to predict the pressure at the array resulting from a $Q_s e^{+i\omega t}$ signal emitted by the pinger and reflected off a target in state x. Define $\mathbf{p}_0(x)$ as the signal (i.e., complex vector of pressures) at the array resulting from the incident signal from the pinger being reradiated from a point source at x with property that magnitude of the reflected pressure at the reference distance (e.g., 1 yard) is equal to the magnitude of the incident pressure at x.

Define $Q(x)$ to be the ratio of the reflected pressure at the reference distance to the incident pressure given the target is in state x. Let $TS(x)$ be the target strength of the target given it is in state x. Then

$$Q(x) = 10^{\frac{TS(x)}{20}}.$$

Now we can write

$$\mathbf{P}_0(x) = Q(x)\,\mathbf{p}_0(x). \tag{66}$$

The signal $\mathbf{p}_0(x)$ is calculated by using the modal decomposition to compute the signal that arrives at the target $P_t(x)$, given the target is in state x and then assuming that this signal is reradiated from the target as though it were a point target with $Q(x) = 1$. The signal received at the array is calculated by applying the modal decomposition to $P_t(x)$ and propagating it to the array.

7.2.2.3 Propagation from Pinger to Target

To calculate the sound propagation from the pinger to the target, we use Eq. (62):

$$P_t(x) = \sum_m \sqrt{\frac{2\pi}{k_m r_t(x)}}\, Q_s U_m(z_p) U_m(z_t(x)) \exp\left(-ik_m r_t(x)\right)$$

where

z_p = depth of the pinger,
$z_t(x)$ = depth of the target in state x
$r_t(x)$ = range of target in state x from pinger
Q_s = pinger strength (measured in pressure at reference distance).

7.2.2.4 Target Reradiation

The target reradiates the sound in a radiation pattern dependent upon the target's alignment and profile. The value of $Q(x)$ reflects this pattern. For the purpose of calculating $\mathbf{p}_0(x)$, we assume $Q(x) = 1$.

7.2.2.5 Propagation from Target to Sensor

The sensor is a vertical array of hydrophones in the water placed at depths z_h for $h = 1,\ldots, n$. Using Eq. (62) with $P_t(x)$ in place of Q_s, we calculate $\rho_h(x)$, the pressure at the hydrophone with depth z_h, as follows:

$$\rho_h(x) = \sum_m \sqrt{\frac{2\pi}{k_m r_a(x)}} P_t(x) U_m(z_t(x)) U_m(z_h) \exp\left(-ik_m r_a(x)\right) \text{ for } h = 1,\ldots, n \tag{67}$$

where

z_h = depth of the hydrophone h,
$z_t(x)$ = depth of the target in state x
$r_a(x)$ = range from target in state x to the array.

Now we have

$$\mathbf{p}_0(x) = (\rho_1(x), \rho_2(x), \ldots, \rho_n(x))^T.$$

Using Eq. (62), we rewrite the probability distribution in Eq. (65) as follows:

$$Pr\{\mathbf{P}|Q(x), \mathbf{p}_0(x)\} = \frac{1}{\left(\sqrt{2\pi}\right)^n \sqrt{|\Sigma|}} \exp\left(-\tfrac{1}{2}(\mathbf{P} - Q(x)\mathbf{p}_0(x))^H \Sigma^{-1} (\mathbf{P} - Q(x)\mathbf{p}_0(x))\right). \tag{68}$$

Equation (68) represents the probability of receiving $\boldsymbol{P}$ given $Q(x)$ and a theoretical unit strength signal return $\mathbf{p}_0(x)$.

We handle our lack of knowledge of $Q(x)$ by finding the target strength Q_{best} that maximizes Eq. (68) for a given $\mathbf{P}$ and $\mathbf{p}_0(x)$. To do this, we rotate and scale the covariance matrix Σ to be diagonal. Since Σ is a covariance matrix, it is positive definite and there is a nonsingular matrix $\mathbf{R}$ that satisfies

$$\mathbf{R}\,\Sigma\,\mathbf{R}^H = \mathbf{I} = (\mathbf{R}\,\Sigma\,\mathbf{R}^H)^{-1}$$

$$\Sigma^{-1} = \mathbf{R}^H\mathbf{R}. \tag{69}$$

We can now find the target strength Q_{best} that maximizes Eq. (68) which is equivalently expressed in Eqs. (70) and (71):

$$Q_{\text{best}}(x) = \left\{Q: -\frac{1}{2}\left(\mathbf{P} - Q(x)\mathbf{p}_0(x)\right)^H \mathbf{R}^H\mathbf{R}\left(\mathbf{P} - Q(x)\mathbf{p}_0(x)\right) \text{ is maximized}\right\} \tag{70}$$

UNCLASSIFIED

$$Q_{\text{best}}(x) = \left\{ Q: -\frac{1}{2}\left(\mathbf{RP} - Q(x)\mathbf{Rp}_0(x)\right)^H \left(\mathbf{RP} - Q(x)\mathbf{Rp}_0(x)\right) \text{ is maximized} \right\}. \tag{71}$$

The target strength $Q_{\text{best}}(x)$ that maximizes $Pr\{\mathbf{P}|\ Q(x),\ \mathbf{p}_0(x)\}$ in Eq. (71) is calculated to be

$$Q_{\text{best}}(x) = \frac{\mathbf{Rp}_0(x)^H \mathbf{RP}}{\left|\mathbf{Rp}_0(x)\right|^2}. \tag{72}$$

7.2.2.6 Likelihood Function

The likelihood function in SWATS uses the maximum likelihood signal strength. The covariance matrix for **P** is assumed to be $\sigma^2\mathbf{I}$, so that the corresponding coordinate transform matrix **R** is $\sigma\mathbf{I}$.

For a given target state x, we calculate $\mathbf{p}_0(x)$ which is the theoretically expected signal for a target with unit target strength. We also calculate $|Q_{\max}(x)|$, which is the maximum possible target strength for target state x. If $|Q_{\text{best}}(x)| > |Q_{\max}(x)|$, we scale down $Q_{\text{best}}(x)$ so that it has a magnitude equal to $Q_{\max}(x)$. The likelihood function L that is used in SWATS comes from Eq. (68) and is calculated within a scale factor as

$$L(\mathbf{P}|x) = \exp\left(\frac{1}{2\sigma^2}\left|\mathbf{P} - Q_m(x)\mathbf{p}_0(x)\right|^2\right) \tag{73}$$

where the maximum likelihood target strength $Q_m(x)$ is calculated as

$$Q_m(x) = Q_{\text{best}}(x) = \frac{\mathbf{p}_0(x)^H \mathbf{P}}{\left|\mathbf{p}_0(x)\right|^2};\ \left|Q_{\text{best}}(x)\right| \le \left|Q_{\max}(x)\right| \tag{74}$$

$$Q_m(x) = \frac{\left|Q_{\max}(x)\right|}{\left|Q_{\text{best}}(x)\right|} Q_{\text{best}}(x);\ \left|Q_{\text{best}}(x)\right| > \left|Q_{\max}(x)\right|. \tag{75}$$

In SWATS, $Q_{\max}(x)$ is obtained as an input.

7.3 Incorporation of Subjective Information

Subjective information can be incorporated into Nodestar in two ways: in the prior distribution on target state and by the use of likelihood functions.

7.3.1 Prior Distribution

The prior distribution on target state contains information about the target's initial location, velocity distribution, and class. All of these can be adjusted on the basis of subject (or objective) information that is available before Nodestar begins to track the target.

7.3.2 Likelihood Functions

Subjective information that is obtained after a track is initiated can be incorporated by using subjective likelihood functions. A simple example will suffice to illustrate how this is done.

Suppose that we have subjective information about the class of one of the targets that we are tracking. If we can quantify the information in terms of subjective probabilities, then we can define a likelihood function and update the target state distributions accordingly.

As an example, suppose that we have K target classes, $\{a_k; k = 1, \ldots, K\}$, in the target state space. We might receive an intelligence report about the target that contains information about the target's class. Suppose that on the basis of this information we assign the following subjective probabilities to the class of the target:

$$Pr\{a = a_k\} = \lambda_k \text{ for } k = 1, \ldots, K. \tag{76}$$

Because information updating by the use of likelihood functions is insensitive to multiplicative factors (see Eq. (13)), the λ_k's need not add to one. In fact, they can be relative credences or weights.

Using Eq. (76) we define the following likelihood function.

$$L(x) = L((z, v, a)) = \lambda_k \text{ for } a = a_k \text{ and } x = (z, v, a) \in \boldsymbol{X}. \tag{77}$$

Let $q(t,\cdot)$, be the probability distribution on the target state at time t. The posterior at time t, given the subjective information on target class, is computed by

$$\tilde{q}(t, x) = \frac{1}{C} L(x) q(t, x) \text{ for } x \in \boldsymbol{X}$$

where

$$C = \int_{\boldsymbol{X}} L(x) q(t, x) dx$$

as given by Eq. (13).

REFERENCES

Anderson, T. W. (1984). *An Introduction to Multivariate Statistical Analysis*, second edition. Wiley, New York.

Bar-Shalom, Y. and Fortman, T. E. (1988). *Tracking and Data Association.* Academic Press, New York.

Curtis, E. C., Richardson, H. R., and Loane, E. P. (1966). *Cumulative Detection Probability for Continuous Parameter Stochastic Processes.* Daniel H. Wagner Associates Report to Naval Air Development Center.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory.* Academic Press, New York.

Jensen, F. B., Kuperman, W. A., Porter, M. B., and Schmidt, H. (1994). *Computational Ocean Acoustics.* American Institute of Physics Press, New York.

Lindley, D. V. (1972). *Bayesian Statistics, A Review.* SIAM, Philadelphia PA.

Loane, E. P., Richardson, H. R., and Boylan, E. S. (1964). *Theory of Cumulative Detection Probability.* Daniel H. Wagner, Associates Report to the U.S. Navy Underwater Sound Laboratory.

Mori, S., Chong, C-Y., Tse, E., and Wishner, R. P. (1986). "Tracking and Classifying Multiple Targets without Apriori Identification." *IEEE Trans. Automat. Contr.* **AC-31** (May) pp. 401-409.

UNCLASSIFIED

Morrison , D. F. (1976). *Multivariate Statistical Methods.* McGraw Hill, New York (Second Edition).

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufman, San Mateo, CA.

Porter, M. B. (1991). *The KRAKEN Normal Mode Program.* SACLANTCEN Memorandum SM-245, SACLANT Undersea Research Center, San Bartolomeo, Italy.

Porter, M. B. and Reiss, E. L. (1984). "A Numerical Method for Ocean-acoustic Normal Modes." *J. Acoustical Society of America* **76** (July) pp. 244-252.

Tolstoy, A. (1993). *Matched Field Processing for Underwater Acoustics.* World Scientific, River Edge, New Jersey.

Waltz, E. and Llinas, J. (1990). *Multisensor Data Fusion.* Artech House, Dedham MA.

UNCLASSIFIED

Appendix A

SPHERICAL MOTION MODEL AND GRID

This Appendix describes the modifications made to Nodestar to change from a flat Earth grid model to a round Earth grid. Section A1 describes the rationale for the changes. Section A2 describes the mathematics involved. Section A3 reviews current practice in applying motion models to flat grids. Section A4 describes the changes we have made so that we can apply motion models to round grids. Section A5 describes the problems we cannot avoid in going to the round Earth model.

A1. RATIONALE

In the past, Nodestar was a single-target tracker. We assumed that we knew approximately where a target was so that we could begin the tracking. Thus, our grids represented only a small portion of the Earth. In fact, even in debugging runs, the grids used in Nodestar never covered more than 10°, and so at most 600 nmi. At the Equator, the amount of distortion of the sides of the grid caused by this is at most 2 percent. For a submarine moving at 10 kt and within 6 h travel from an island, the grid is 2° by 2°, and the distortion is at most 0.01 percent.

Because Nodestar is now a multiple-target tracker designed for problems involving large areas of the ocean, it must consider the Earth's curvature. Fortunately, the mathematics of navigation leads a simple modification to the methods used for a flat Earth. We can find the equations of motion for a ship traveling along a constant bearing, and our results are nearly the same as the results we had for a flat Earth.

A2. THE MATHEMATICS OF NAVIGATION

First, we give some definitions. The spherical coordinates for the surface of the Earth are ϕ and θ, where ϕ is the *colatitude*, defined as 90° minus the latitude, and θ is the longitude. Let us define φ to be the latitude, so that $\varphi = 90° - \phi$. Since the metric for the sphere in spherical coordinates is given by Eq. (A1) below, the metric in latitude-longitude coordinates is given by Eq. A2 below.

$$ds^2 = d\phi^2 + \sin^2 \phi \, d\theta^2, \tag{A1}$$

$$ds^2 = d\varphi^2 + \cos^2 \varphi \, d\theta^2. \tag{A2}$$

A curve described by the motion of a ship maintaining a constant bearing with respect to north is called a *rhumb line* or a *loxodrome*. This is not a great circle, although for short distances it is close. A loxodrome spirals into the north or south pole. We will derive its equation below.

Assume that a ship starts off at latitude φ_0 and longitude θ_0, and that its equation of motion is $\varphi = \varphi(t)$, $\theta = \theta(t)$ in such a manner that its bearing with respect to north is α and its speed is the constant s. Then, using the metric tensor for the latitude-longitude coordinate system, we can use the equation $\mathbf{u}\cdot\mathbf{v} = |\mathbf{u}|\,|\mathbf{v}| \cos \alpha$, where $u = (1, 0)$ and $\mathbf{v} = (\varphi', \theta')$ to find the equation

$$\varphi' = s \cos a. \tag{A3}$$

To find θ', we must use the fact that the length of **v** is s to get

$$\varphi'^2 + \cos^2 \varphi\, \theta'^2 = s^2 . \tag{A4}$$

Substituting Eq. (A3) into Eq. (A4), we get

$$\cos^2 \varphi\, \theta'^2 = s^2 \sin^2 \alpha. \tag{A5}$$

Therefore, we get the following for θ':

$$q¢ = s \sin a \sec j. \tag{A6}$$

Thus, the rate of change of latitude is the same in the round Earth tracker as in the flat Earth tracker, and the distortion in the rate of change of longitude is quite simple. Incidentally, we can get a parametric solution for ϕ and θ by substituting for s in Eq. (A6):

$$q¢ = j¢ \tan a \sec j, \tag{A7}$$

$$q¢ \cot a = j¢ \sec j, \tag{A8}$$

$$\theta \cot \alpha = \ln \tan \left(\frac{\varphi}{2} + \frac{\pi}{4}\right) + c. \tag{A9}$$

A3. FLAT EARTH GRIDS

Originally, Nodestar used a simple model for moving target positions within a given grid and motion model. It assumed that the probability densities for a grid cell are concentrated at the cell center as in point A in Fig. A1. If we have a certain velocity under consideration, (v_x , v_y), and the grid cell size is (dx, dy), then the target will move one cell to the right at time $t_x = dx/v_x$ and one cell up at time $t_y = dy/v_y$. It will move further to the right or up at every multiple of t_x or t_y, respectively.

Quite simply, we keep track of the current time, t_x , and t_y , and we move the probabilities for all cells in lockstep at the multiples of t_x and t_y. Since these parameters do not depend at all on the original cell indices, cells will never overlap. The target traveling from A to B below, in our model, enters the cells in the manner described below.

A target starting at point A traveling toward point B was assumed to go first to the right, then up, then right again. Each time that it passed a dashed line, denoting that it passed the center of a grid box either horizontally or vertically, it jumped a box in the internal representation. The model would do the same for any target traveling parallel to AB.

A4. THE CONVERSION TO ROUND EARTH GRIDS

The difference between round Earth grids and flat Earth grids is that the areas of different grid boxes will be different. The area of the grid box with latitude going from φ to $\varphi+\Delta\varphi$ and longitude going from θ to $\theta+ \Delta\theta$ is $[\sin (\varphi+\Delta\varphi)- \sin \varphi] \times \Delta\theta$. This last figure is approximately equal to $\cos\varphi \times \Delta\varphi\ \Delta\theta$, by the mean value theorem. The lengths of the north and south sides of the grid box are related similarly.

UNCLASSIFIED

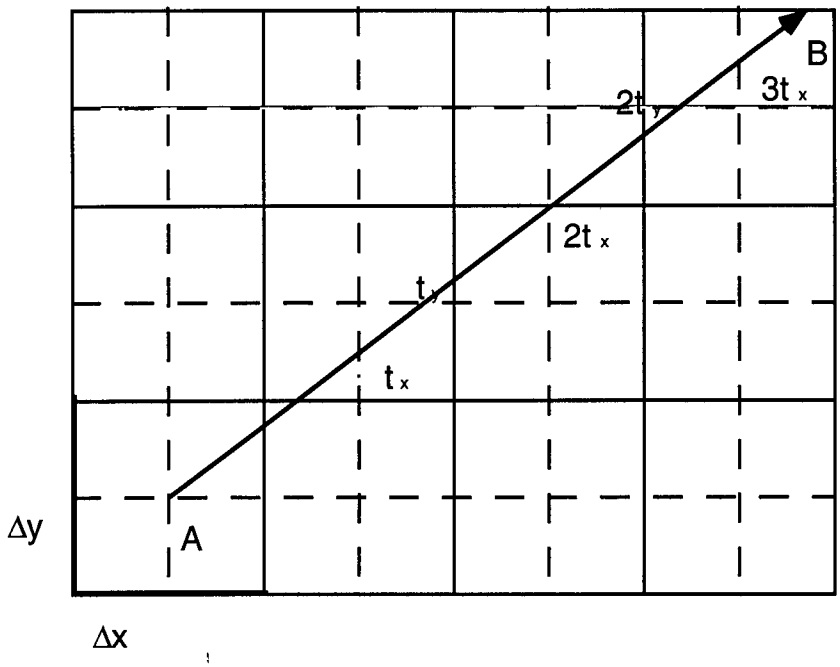


Fig. A1 — Flat Earth grid

Fortunately, targets change their latitudes in this model at the same times as in the previous model by Eq. (A3) above. That equation is the same as the analogous equation in the flat model. For the spherical model, Nodestar no longer treats every cell in the same way. It does treat each row the same way. In other words, Nodestar keeps track of which row each target position was in at time 0, as follows.

Define the local advancement times for a velocity model for a target traveling at speed s and bearing α in a grid box centered at (φ, θ) with subgrid sizes $(\Delta\varphi, \Delta\theta)$ as follows:

$$t_\varphi = \frac{\Delta\varphi}{s \cos\alpha} \tag{A10}$$

$$t_\theta = \frac{\Delta\theta}{s \sin\alpha} \cos\varphi. \tag{A11}$$

At times t_θ, $2t_\theta$, $3t_\theta$, ..., Nodestar moves the target east. At times t_φ, $2t_\varphi$, 3_φ, it moves the grid box north, and then corrects for the shortening of lines of latitude. Suppose that at time $t = t_\varphi$,

$$0 < t_\varphi < t_\theta \ . \tag{A12}$$

As the target enters the grid box centered at $(\varphi + \Delta\varphi, \theta)$, the value of t_θ will change to

$$t_{\theta new} = \frac{\Delta\theta}{s \sin\alpha} \cos(\varphi + \Delta\varphi). \tag{A13}$$

To smooth out the transition a bit, let us average the t_θ's by integration. We integrate the expression for t_θ with respect to φ, and we get

$$t_{\theta avg} = \frac{1}{\Delta\varphi} \int_{\varphi}^{\varphi+\Delta\varphi} \frac{\Delta\theta}{s \sin\alpha} \cos t dt \tag{A14a}$$

$$= \frac{\Delta\theta}{s \sin\alpha} \times \cos(\varphi + \Delta\varphi/2) \frac{\sin\Delta\varphi/2}{\Delta\varphi/2} \tag{A14b}$$

$$\approx \frac{\Delta\theta}{s \sin\alpha} \times \cos(\varphi + \Delta\varphi/2). \tag{A14c}$$

When a target goes up a box, look at the time that has passed since it has moved horizontally. Call this time t_h. If t_h is greater than $t_{\theta avg}$, we move the target right and subtract $t_{\theta avg}$ from t_h. In any case, we multiply t_h by $t_{\theta new}$ divided by $t_{\theta avg}$ and continue. Consider the diagram in Fig. A2.

For a target traveling from A to D, Nodestar moves it up one grid line at point B. The correction moves it to the right, and then it continues to C and D. Similarly, a target moving from E to I has a correction made at point G.

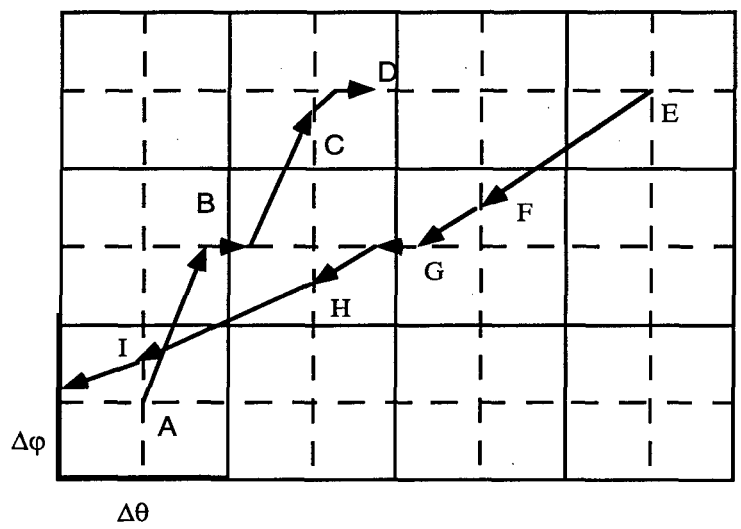


Fig. A2 — Motion in round Earth grid

A5. UNAVOIDABLE DIFFICULTIES IN ROUND MOTION MODELS

One might think that the problems we face in designing homogeneous grids for the round Earth model simply indicate that we have not adequately thought through our problems. This, however, is false. No matter what we do, we cannot find equal-area, translation-invariant grids for the sphere. This is a matter of algebraic topology, not of analysis.

Suppose we could find such a grid. Then, we would have coordinates for the sphere without any singularities. We could get a nonvanishing continuous vector field on the sphere. This, in fact, is impossible. By the Poincaré-Hopf Theorem (see Guillemin and Pollack (1974))*, any continuous vector field must have at least one zero element.

* Guillemin, V. and Pollack, A. (1974). *Differential Topology*. Prentice Hall, Englewood Cliffs, New Jersey.

UNCLASSIFIED

This result means that any grid scheme must break down at least one point. We could create an artificial coordinate system on the sphere where there was one pole in the middle of a large land mass, but we would need to be careful to have this be part of the internal implementation only.

UNCLASSIFIED

Appendix B

A SUMMARY OF THE COMPARISON OF LINEAR AND NONLINEAR TRACKERS

This report presents the results of a series of tests designed to compare the performance of the discrete nonlinear tracker, Nodestar, to that of an extended Kalman filter in a variety of situations involving the use of line-of-bearing detections to track a submarine. The extended Kalman filter used in this comparison is the Maneuvering Target Statistical Tracker (MTST). MTST is used in a number of Navy systems including the OBU tracker and NTSA's TIMS/MRS track reconstruction system (see Gerr (1984).)

Thirty cases were run. In each case, there were three fixed passive arrays and a target track that was chosen in a random fashion. Detection of the target by the arrays was simulated using an l-s process for acoustic fluctuations. (The l-s process is defined in Loane et al. (1964).) When a detection occurred, the simulation calculated the true bearing of the target-to-detecting sensor and then added a random error. Each case lasted for 20 hours of simulated time. For each case, the file of detections and noisy bearings was recorded. The same file was fed to Nodestar and MTST. Every 15 minutes during the run, each tracker produced an estimate of the target's location in terms of a probability distribution. These estimates were recorded and used to calculate measures of effectiveness (MOEs) that measure the performance of the two trackers.

B1. DESCRIPTION OF MOEs

Both Nodestar and MTST are statistical tracers in the sense that their estimates of target position are in terms of probability distributions. MTST's output is displayed as ellipses. The ellipses represent bivariate normal distributions for target location and are the contours of the 86 percent containment region. Nodestar's output is a discrete probability distribution on target state. Each cell in the distribution is color-coded to indicate its probability.

We used three MOEs:

AOU Size. This is the area of the smallest region containing 86 percent of the probability distribution on target location. For MTST this is the area of the 86 percent (2-s) ellipse.

Mean Missed Distance (MMD). For this measure we must know the target's actual position. We then calculate MMD as the mean squared distance of the target's actual position from the probability distribution on target location. MMD, then, is the generalization to probability distributions of the mean missed distance that one would calculate if the tracker gave a point estimate for target position.

Accuracy. We measure the accuracy of the probability distributions produced by a statistical tracker in terms of its containment regions. As an example, take the 86 percent containment region for MTST (i.e., the ellipses). Suppose we plotted the actual target position at 100 times and compared it to the 86 percent region at those times. We want the target's position to fall inside the

region 86 percent of the time and outside the remaining 14 percent. Similarly, if we looked at the 50 percent region, we would want the target to be in 50 percent of the time and out the other 50 percent. If this is true, then our distribution is giving us an accurate representation of the degree to which we have localized the target.

Accuracy is a measure of the extent to which a tracker's containment regions are an accurate representation of its knowledge of the target's location. Accuracy varies from 100 percent to 0 percent.

In making our comparison, we calculated the AOU size and MMD at each time the trackers made an estimate (i.e., 81 times for each case) and computed the average AOU size and MMD as our comparison statistics. The accuracy MOE is computed for an entire case. It is based on the distributions produced at the 81 times during the run.

B2. ROBUSTNESS TESTING

Both Nodestar and MTST require the user to make probabilistic assumptions about the target's motion. Since Nodestar uses detection/no detection information from the arrays to help estimate the target's location, it requires an estimate of the figure of merit (FOM) for the target's primary (i.e., most detectable) frequency and the propagation loss curve for this frequency. This acoustic information is necessary for the detection/no detection likelihood functions calculated by Nodestar. We investigated the robustness of tracker performance when there is a mismatch between the assumptions used by the tracker (i.e., the motion and acoustic assumptions) and those used in the simulation that generated the file of detections fed into the trackers.

B3. TEST SCENARIOS

We used the following scenarios in our tests.

Target Motion Models. For the target motion in the simulator, we used three basic types of models: patrolling, constant-course-and-speed, and evading. We used the three base tracks shown in Figs. B1 through B3. The figures also show the locations of the three fixed array sensors (labeled 1, 2, and 3) used to generate detections in the simulation. The rectangle in each figure marks the end of the base track.

Patroller. For the 10 patroller cases, we used the base track shown in Fig. B1. The speeds for each of the legs are given in Table B1. For each case, we perturbed the length of each leg by making a random draw to adjust the length of the leg.

Constant-Course-and-Speed. For the 10 constant-course-and-speed cases we used the target track shown in Fig. B2 with a speed of 12 kt.

Evader. Figure B2 shows the base track for the evader motion. Table B2 gives the speeds for each of the legs. For each case, we perturbed the length of each leg by making a random draw to adjust the length of the leg.

UNCLASSIFIED

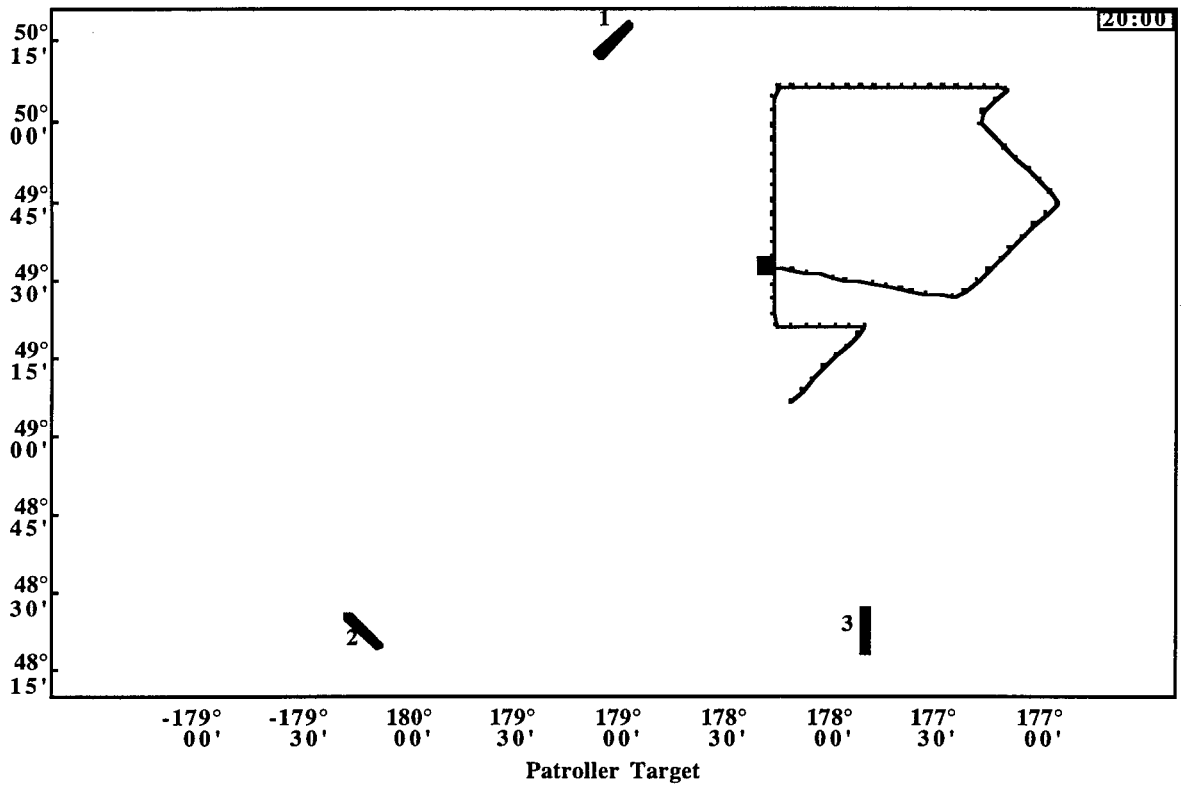


Fig. B1 — Patroller base track

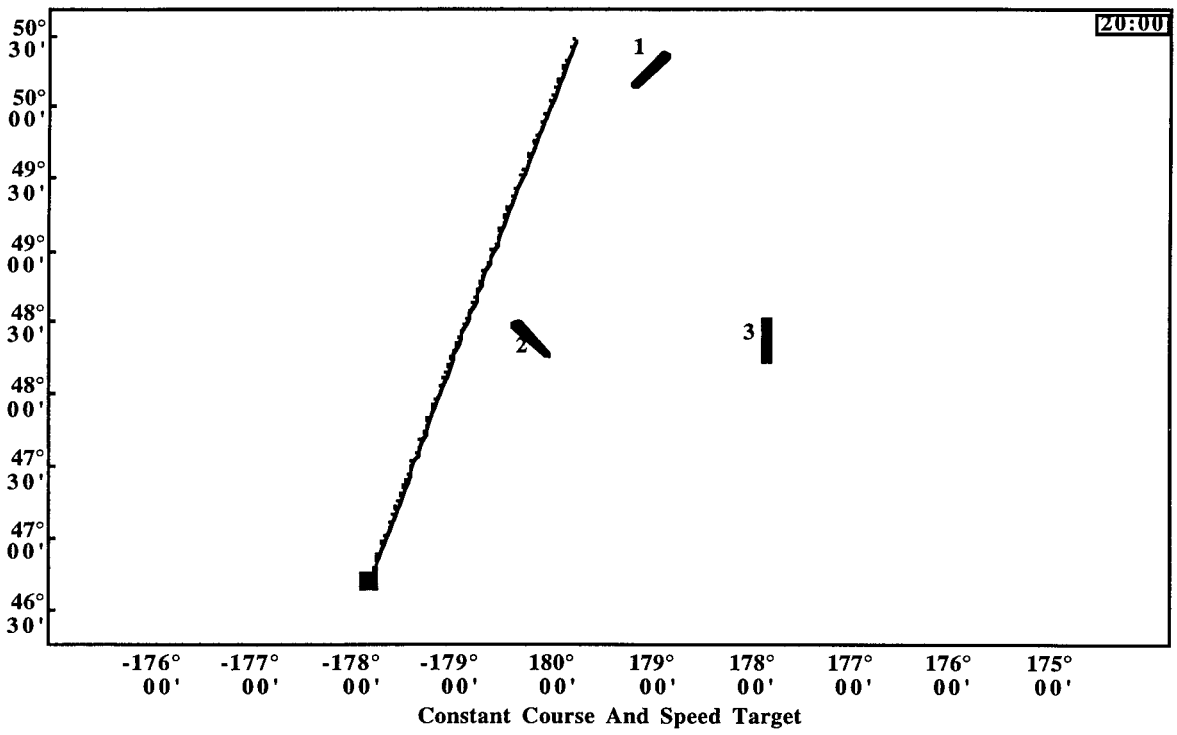


Fig. B2 — Constant course and speed base track

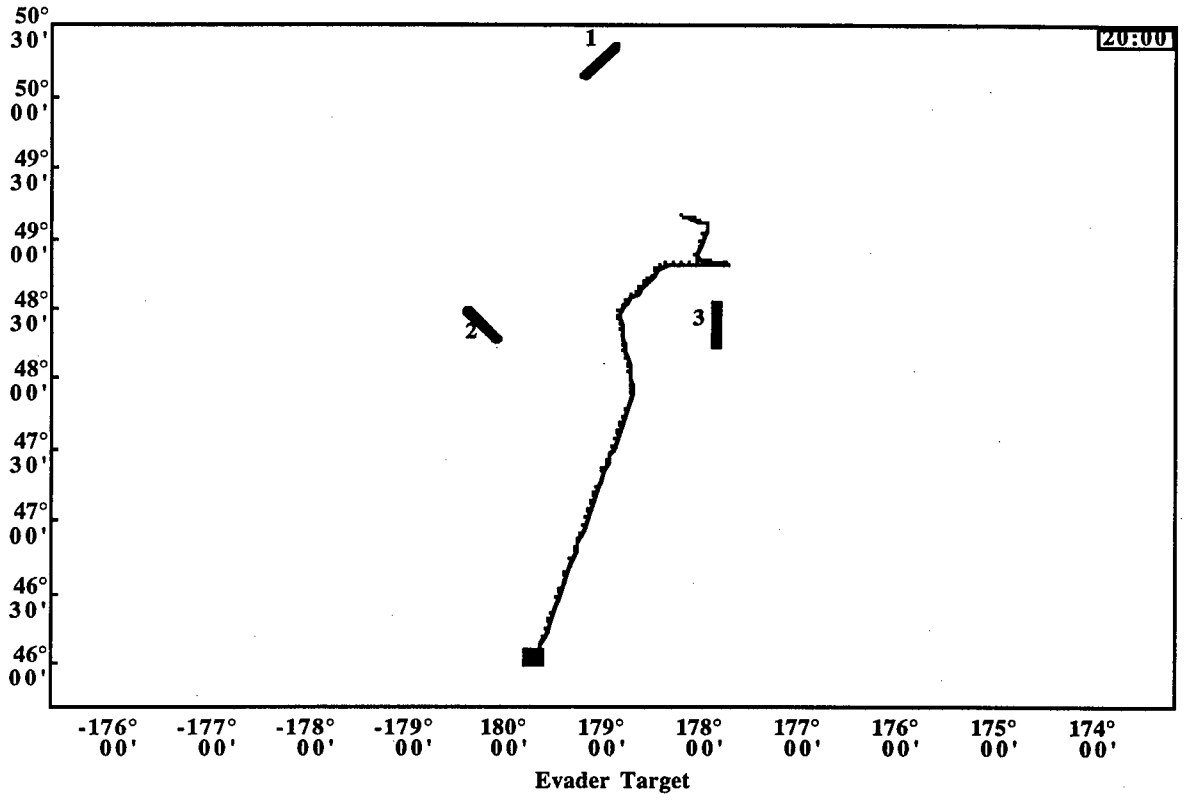


Fig. B3 — Evading target base track

Table B1 — Patroller Tracks

Leg	Course (deg)	Speed (kt)	Time on Leg (hr) *
1	45	12	RAND(2.5)
2	270	11	RAND(2.0)
3	0	11	RAND(4.0)
4	90	10	RAND(3.0)
5	225	12	RAND(1.0)
6	135	12	RAND(1.5)
7	225	12	RAND(3.0)
8	280	10	RAND(5.5)

*Note: RAND(x) indicates a random draw from a uniform distribution over [x/2, 3x/2]. The base case has time on leg equal to x hr.

UNCLASSIFIED

Table B2 — Evader Tracks

Leg	Course (deg)	Speed (kt)	Time on Leg (hr) *
1	110	10	RAND(1.0)
2	200	12	RAND(1.0)
3	100	10	RAND(1.0)
4	270	13	RAND(1.5)
5	220	10	RAND(2.0)
6	170	12	RAND(2.5)
7	200	14	RAND(12.0)

*Note: RAND(x) indicates a random draw from a uniform distribution over [x/2, 3x/2]. The base case has time on leg equal to x hr.

Tracker Motion Assumptions. Both Nodestar and MTST used one fixed motion model for all 30 test cases. The Nodestar motion model assumed that the target's heading was uniform [0, 360°] and its speed distribution uniform over [5,20 kt]. The target course changes were modeled as taking place at exponentially distributed times with a mean time of 1 hour between course changes.

The motion model in MTST is an Integrated Ornstein-Uhlenbeck process. The choice of the parameter values for this motion model was made in conformance with the recommendations in Gerr (1984). The objective was to match the motion assumptions used by Nodestar as closely as possible.

Acoustic Assumptions. The test considered three acoustic environments: surface duct, direct path and bottom bounce, and convergence zone.

Nodestar requires that the user specify a propagation loss curve and an FOM for the target. To test the robustness of Nodestar to mismatches in the propagation loss curve and the FOM, we considered the following cases:

1. Correct transmission loss curve, correct estimate of FOM.
2. Correct transmission loss curve, high estimate of FOM.
3. Correct transmission loss curve, low estimate of FOM.
4. Incorrect transmission loss curve, correct estimate of FOM.

We ran one test for each combination of assumptions listed above. This produced three motion models × three propagation loss curves × three FOM assumptions + three mismatched propagation loss curves = 30 runs.

Each run was given a three-letter descriptive label. Table B3 describes the meaning of the three-letter label.

Table B3 — Run Labels

First Letter	Second Letter	Third Letter
Acoustic Environment	Simulated Target Motion	Acoustic Information Accuracy (Mismatch)
C = Convergence Zone	P = Patroller	C = Correct Acoustic Assumption
D = Direct Path	C = Constant Course and Speed	H = High estimate of FOM
S = Surface Duct	E = Evader	L = Low estimate of FOM
		I = Incorrect Propagation Loss

B4. RESULTS OF COMPARISON

Table B4 summarizes the results of the 30 runs. Each line of the table shows the values of the three MOEs (AOU size, mean missed distance, and accuracy) that were computed for Nodestar and MTST for the indicated run. Each run covered a period of 20 hours. Every 15 minutes, each tracker produced an estimate of the target's location for a total of 81 times at which estimates were produced (including the initial estimate at time 0). The better of the two numbers in each category for a run is labeled with an asterisk. A quick glance shows that Nodestar wins most of the comparisons. The columns labeled G and H give the number of times at which contact was gained and held respectively during a run. The sum of the gains and holds yields the total number of contacts made by all three arrays during the run.

The first row of Table B4 shows the results for the case of a convergence zone environment, constant-course-and-speed target, and correct estimates by the Nodestar of the FOM and propagation loss curve. Note that Nodestar has a lower average AOU size, smaller MMD, and higher accuracy than MTST. The next two lines show the results for two cases with the same target motion and environmental characteristics as the first, but with Nodestar assuming that the FOM is one standard deviation higher and lower respectively than it actually is in the simulation. Even with this mismatch in acoustic assumptions, Nodestar wins in all three categories.

One would expect Nodestar to outperform a Kalman filter such as MTST in a convergence zone environment. The results in Table B4 bear that out. In 27 comparisons, there are only three instances in which MTST has a better MOE than Nodestar. Overall, Nodestar is clearly superior.

In the case of a direct path or ducting environment, one might expect that Nodestar's edge over MTST would diminish because the acoustic information in this environment would not be as valuable as in a convergence zone environment. Surprisingly, the results in Table B4 show that Nodestar continues to outperform MTST in direct path and surface duct environments even in the face of mis-estimates of FOM. Examining the three runs (DCI, DEI, and DPI) in which the propagation loss curve was convergence zone but Nodestar thought it was direct path, we see that Nodestar bettered MTST in eight of the nine MOE comparisons.

UNCLASSIFIED

Table B4 — Summary Results

Run	AOU (nm^2)		MMD (nm)		Accuracy (%)			
	Nodestar	MTST	Nodestar	MTST	Nodestar	MTST	G	H
CCC	2,839*	4,880	36.83*	64.76	69.76*	27.62	4	13
CCH	2,354*	4,158	33.47*	54.91	79.91*	33.36	5	16
CCL	946*	1,122	23.41*	28.74	70.19*	45.18	10	49
CEC	1,571*	2,530	25.62*	33.39	67.19*	65.17	6	52
CEH	2,874*	5,822	41.61*	69.11	52.34	53.88*	7	8
CEL	2,908*	4,641	46.84*	46.94	46.38	59.28*	4	23
CPC	4,597*	7,584	41.07*	45.31	67.58	90.45*	3	9
CPH	4,017*	7,247	32.59*	49.24	88.3*	65.90	3	19
CPL	11,080*	16,860	56.83*	65.19	84.48*	81.9	2	0
DCC	1,057*	1,743	21.73*	48.21	50.7*	30.39	5	32
DCH	4,852*	8,849	66.21*	74.38	19.04	41.13*	2	7
DCL	1,911*	3,620	36.07*	58.60	25.61	27.99*	3	26
DCI	1,482*	2,595	21.45*	50.62	63.57*	36.42	6	25
DEC	3,705*	6,688	40.86*	63.76	71.65*	61.95	3	5
DEH	2,012*	3,697	24.07*	32.63	71.41	79.19*	4	29
DEL	3,768*	6,147	66.64*	96.15	30.05*	25.37	2	13
DEI	4,664*	8,129	66.68*	70.99	27.02	44.83*	4	5
DPC	898*	1,322	21.55*	28.90	70.83*	68.99	9	24
DPH	2,155*	3,831	36.98*	41.28	41.42	70.93*	6	16
DPI	2,670*	4,706	26.82*	39.80	72.15*	61.60	4	26
DPL	6,680*	10,940	53.20*	60.94	62.53*	58.07	2	0
SCC	5,668*	8,162	72.26	63.92*	39.96	54.45*	7	3
SCH	1,224*	1,529	22.34*	39.37	83.21*	40.36	8	30
SCL	5,755*	9,273	70.14*	75.94	53.64*	43.45	3	4
SEC	24,920	21,800*	181.50	133.10*	9.77	32.00*	0	0
SEH	3,275*	5,139	30.06*	65.81	89.13*	50.01	4	22
SEL	3,160*	4,410	41.55*	45.08	65.58*	65.25	5	16
SPC	1,750*	2,587	27.43*	34.01	71.12*	70.91	8	13
SPH	1,639*	2,310	29.66*	40.33	62.48*	50.30	6	20
SPL	860*	1,261	15.97*	33.90	89.47*	43.12	11	22

* indicates the winner.

Figures B4 through B6 graphically present the comparisons in the table. The comparison of AOU sizes is shown in Fig. B4. The cases are labeled along the horizontal axis. For each case, a bar shows the difference between the AOU size for Nodestar and MTST. The height of the bar indicates the magnitude of the difference. A black bar indicates that Nodestar had a smaller AOU size than MTST. A white one indicates that MTST had a smaller AOU size. One can see that Nodestar beat MTST in all cases except one.

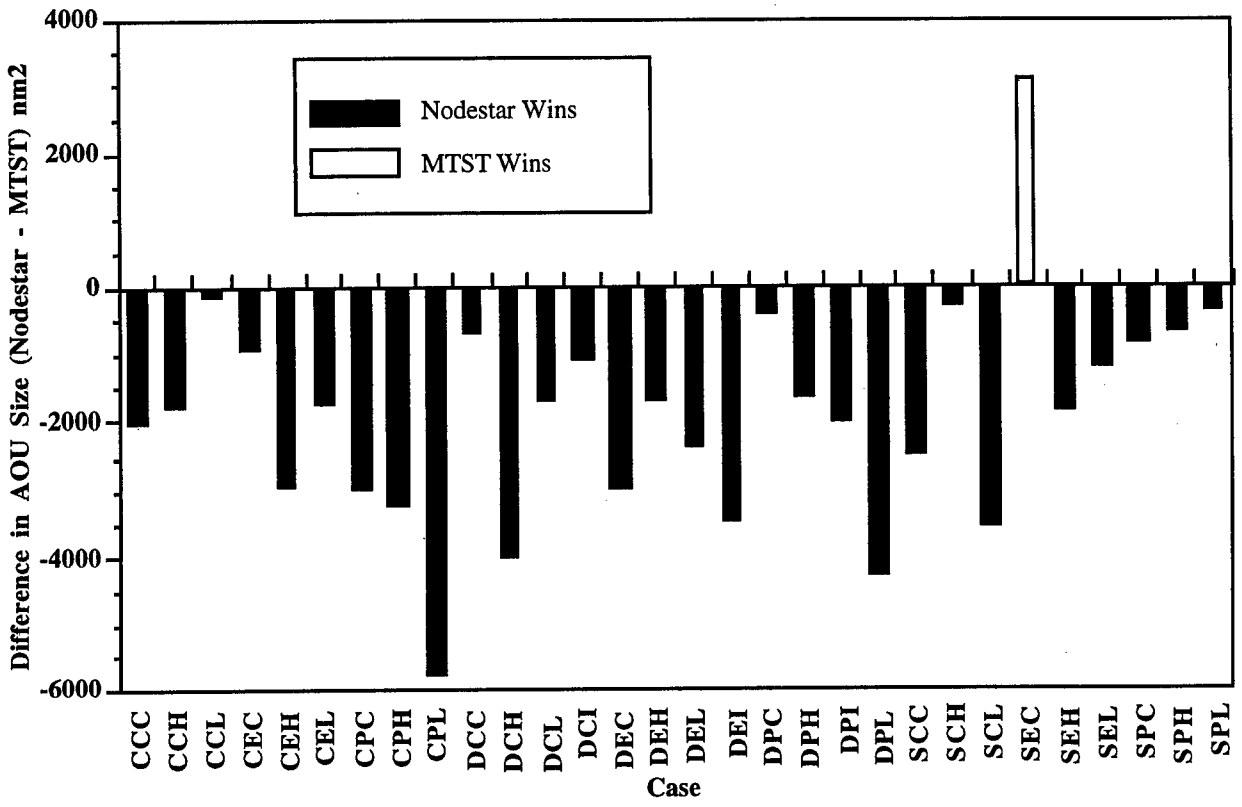


Fig. B4 — AOU comparison

Figure B5 shows the mean missed distance comparison. Nodestar beats MTST in all cases except two. Figure B6 shows the accuracy comparison. Nodestar wins 21 out of 30 of these comparisons.

B5. CONCLUSIONS

The obvious conclusion is that Nodestar performs much better than MTST in favorable situations (e.g., convergence zone environments with correct acoustic estimates) and continues to outperform MTST in cases where Nodestar is given a poor estimate of FOM or even the wrong propagation loss curve. The conclusion was tested statistically using a Wilcoxen test for paired observations. This test showed that Nodestar performed significantly better than MTST for all three MOEs.

One might ask whether the difference between Nodestar and MTST could be attributed to the possibility that MTST is not a good implementation of a Kalman filter. In our opinion, the answer to this question is clearly no. MTST is one of the best Kalman filters for submarine tracking. It has a motion model that provides sensible velocity distributions and it handles linear information in a reasonable fashion given the linear constraints imposed by the Kalman methodology.

UNCLASSIFIED

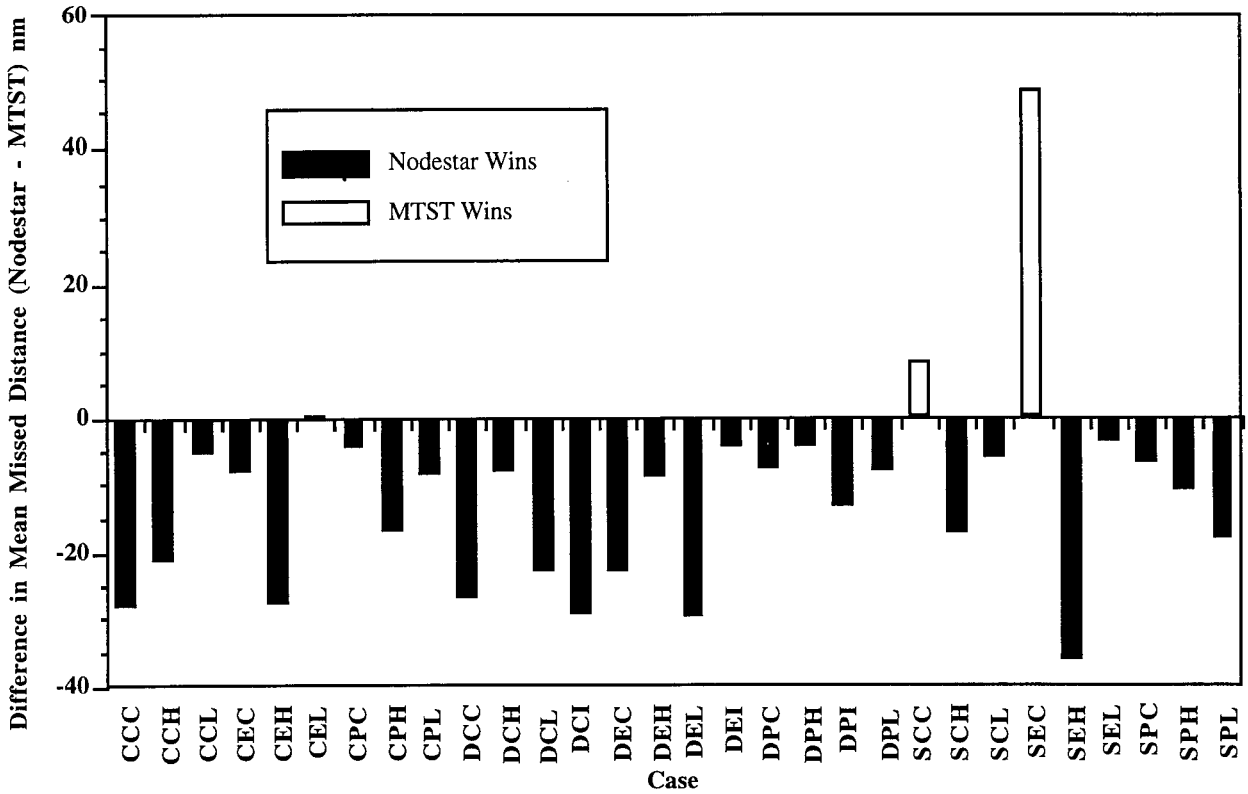


Fig. B5 —Mean missed distance comparison

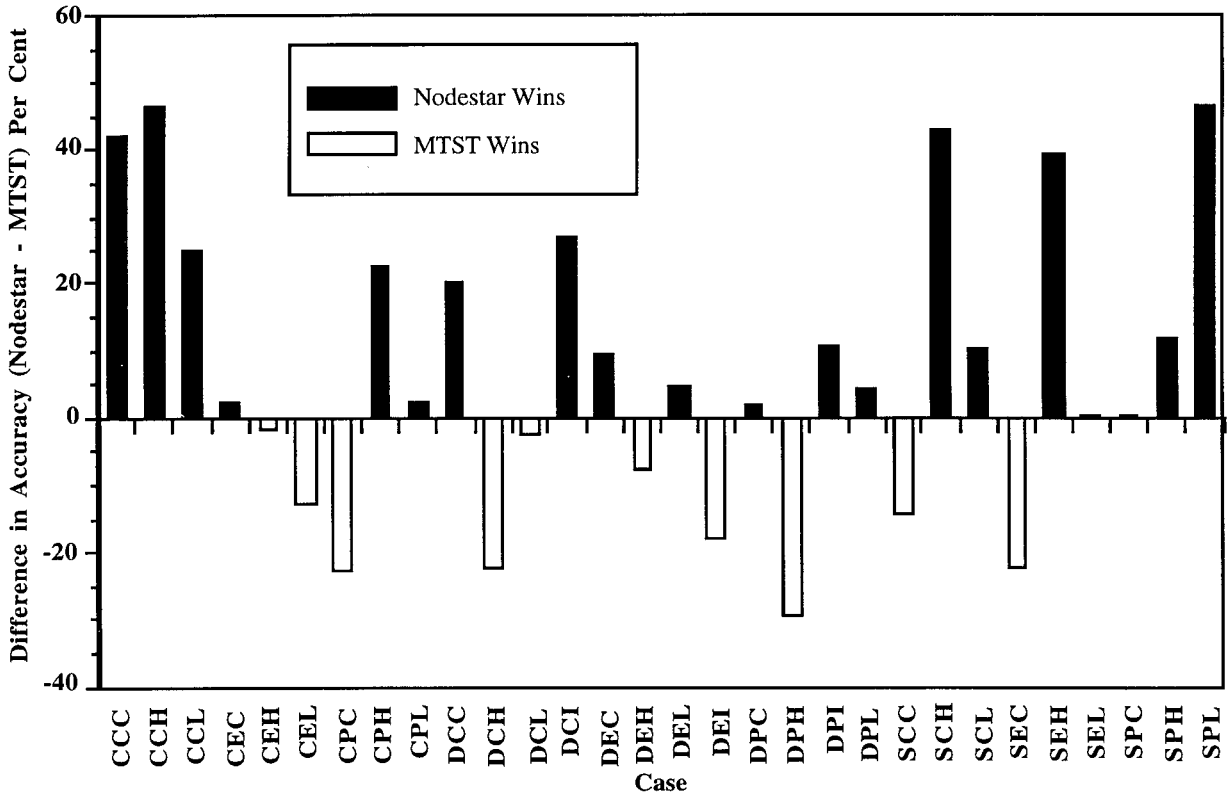


Fig. B6 —Accuracy comparison

When the output from Nodestar and MTST are viewed side by side at each time period, the reasons that MTST performs so poorly compared to Nodestar become clear. First, the solution for range information in the bearing detections used by MTST makes it difficult for MTST to localize the target. The linear approximations required to incorporate line-of-bearing detections cause the target distribution to get drawn into the sensor in a sort of fatal attraction. Inability to use negative information allows the target distribution to expand into regions over long periods when no contact is obtained. All of these problems will be shared by any Kalman filter. They are not specific to MTST.

REFERENCES

Gerr, N. L. (1984). *Technical Documentation for the TIMS/MRS Implementation of the Maneuvering Target Statistical Tracker (MTST) Revision 7.0 (T/M).* Daniel H. Wagner Associates Memorandum to Naval Tactical Support Activity (NTSA).

Loane, E. P., Richardson, H. R., and Boylan, E. S. (1964). *Theory of Cumulative Detection Probability.* Daniel H. Wagner, Associates Report to the U.S. Navy Underwater Sound Laboratory.