



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1997-09

A computationally efficient and cost effective
multisensor data fusion algorithm for the
United States Coast Guard Vessel Traffic
Services system

Midwood, Sean A.

Monterey, California. Naval Postgraduate School

<http://hdl.handle.net/10945/31943>

Downloaded from NPS Archive: Calhoun



Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



THESIS

**A COMPUTATIONALLY EFFICIENT AND COST
EFFECTIVE MULTISENSOR DATA FUSION ALGORITHM
FOR THE UNITED STATES COAST GUARD VESSEL
TRAFFIC SERVICES SYSTEM**

by

Sean A. Midwood

September 1997

Thesis Advisor:
Second Reader:

Murali Tummala
Roberto Cristi

Approved for public release; distribution is unlimited.

19980106 021

DTIC QUALITY INSPECTED 4

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 instruction, 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 September 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE A COMPUTATIONALLY EFFICIENT AND COST EFFECTIVE MULTISENSOR DATA FUSION ALGORITHM FOR THE UNITED STATES COAST GUARD VESSEL TRAFFIC SERVICES SYSTEM			5. FUNDING NUMBERS	
6. AUTHOR(S) Sean A. Midwood				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) This thesis develops an algorithm to fuse redundant observations due to multiple sensor (type and location) coverage in order to provide a significant reduction in duplicate track information provided to Vessel Traffic Services (VTS) operator displays. The design of the algorithm allows acceptance of inputs from any type of sensor (radar, acoustic, GPS, system generated and manual tracks) as long as the basic decision criteria elements are provided. The result of this effort is a computationally efficient and cost effective software solution to a significant system deficiency that impacts greatly on overall waterway safety. The algorithm is tested with real data collected from the VTS system at Puget Sound in September 1996. The results indicate that the algorithm correctly fuses redundant sensor observations on the same vessel resulting in a significant reduction in the amount of unnecessary information presented to the VTS operator.				
14. SUBJECT TERMS Vessel Traffic Services, Redundant Observations, Algorithm Fuse Redundant Observations, VTS, Software			15. NUMBER OF PAGES 77	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18 298-102

Approved for public release; distribution is unlimited.

**A COMPUTATIONALLY EFFICIENT AND COST EFFECTIVE MULTISENSOR
DATA FUSION ALGORITHM FOR THE UNITED STATES COAST GUARD VESSEL
TRAFFIC SERVICES SYSTEM**

Sean A. Midwood
Lieutenant Commander, Canadian Armed Forces
B. Eng., Lakehead University, 1989


Submitted in partial fulfillment
of the requirements for the degree of

**MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL
September 1997**

Author:


Sean A. Midwood

Approved by:


Murali Tummala, Thesis Advisor


Roberto Crista, Second Reader



Herschel H. Loomis, Jr., Chairman
Department of Electrical and Computer Engineering

TABLE OF CONTENTS

I. INTRODUCTION	1
A. GOAL OF THE THESIS.....	2
B. THESIS OUTLINE.....	2
II. THE VTS ENVIRONMENT.....	5
A. RADAR TRACKS	8
B. AUTOMATED DEPENDENT SURVEILLANCE (ADS) TRACKS	9
C. STANDARD ROUTE (SR) TRACKS	12
III. DATA COLLECTION, FORMATTING AND PREPROCESSING.....	13
A. DATA COLLECTION	13
B. FORMATTING	15
C. PREPROCESSING.....	16
IV. FUSION ALGORITHM.....	19
A. WINDOWING OF THE AVAILABLE DATA.....	20
B. MULTISENSOR DATA FUSION	21
1. Fuzzy Association	21
2. Levels Of Fusion	22
C. POSITIONAL FUSION	22
1. Sensor Level	22
2. Central Level	23

LIST OF FIGURES

1. VTS High Level Architecture.....	1
2. Overview of Fusion Algorithm.....	3
3. Current JMCIS Flow for VTS	6
4. JMCIS Software Architecture.....	7
5. Proposed ADS Segment.....	10
6. Test Area	14
7. Overview of Fusion Algorithm.....	19
8. A Flow Chart of the Interpretative Fusion Algorithm.....	24
9. Membership Functions: (a) positional attributes - longitude and latitude; (b) course in degrees; and (c) speed in knots	26
10. Depiction of the Fuzzy Associative Decision System	27
11. Scenario Plot Example: (a) no fusion applied and (b) fusion applied	29
12. Overlapping Radar Coverage On A Single Vessel: (a) no fusion applied and (b) fusion applied	36
13. Overlapping Radar Coverage On A Single Vessel With An Independent Vessel: (a) no fusion applied and (b) fusion applied	37
14. Overlapping Radar and ADS Coverage on a Single Vessel: (a) no fusion applied and (b) fusion applied	38
15. Overlapping Radar and ADS Coverage on Multiple Tracks: (a) no fusion applied and (b) fusion applied	39

LIST OF TABLES

1. Important Variable Names	45
-----------------------------------	----

I. INTRODUCTION

The United States Coast Guard (USCG) operates and maintains the Vessel Traffic Services (VTS) System that manages marine traffic in the major harbors and waterways in the continental USA. The software that controls this system allows for numerous sensor types, predominantly radar, to report on vessels transiting the area of concern to a Vessel Traffic Center (VTC). The VTS can be thought of as the equivalent of an air traffic control system for US waterways. A depiction of the current VTS architecture is shown in Figure 1. At a given VTC, the sensor reports are plotted as tracks on a display, layered over raw radar video, which is used by system operators to provide advisories to vessels that promotes a safe and efficient operating environment. The version of software currently

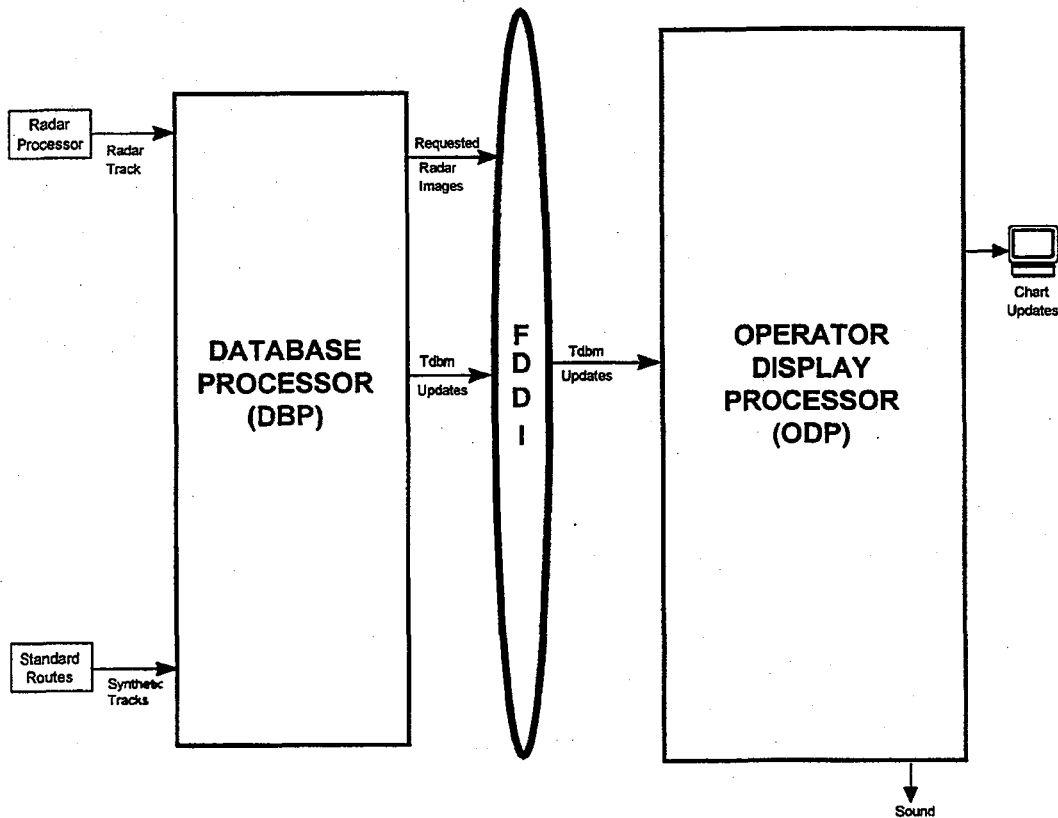


Figure 1. VTS High Level Architecture

employed by the VTS is not capable of correlating redundant reports on the same vessel that are provided by the various sensors in the system. These duplicate tracks, which appear on the VTS displays, are a significant system deficiency that detracts from an operator's ability to manage overall waterway safety.

A. GOAL OF THE THESIS

This thesis presents a proof-of concept algorithm that will perform multisensor data fusion on the sensor information currently provided on vessels in a VTS System. The results are output as a unique set of tracks to an operator display and archived. The approach taken in developing this algorithm is based on the research and findings reported in Glenn [Ref. 1] and Ruthenberg [Ref. 2]. While these works reported primarily on overlapping radar coverage, the proposed algorithm is designed from the outset with radar, GPS based and system generated sources in mind. The algorithm is meant to be able to take data from any available sensor that can provide the necessary attributes in order to make a fusion decision as depicted in Figure 2. As was not the case in the previous work [Ref. 1], actual data from an operational VTS System was obtained which negated the need to simulate vessel traffic. This greatly enhanced the testing and development of the algorithm.

The data is obtained from a variety of sensors which includes radar, ADS and synthetic or computer generated tracks. The radars tracks are provided by commercially available radar sets. ADS tracks are GPS based information sent automatically via radio link from the vessel. Standard Routes (SR) tracks are synthetically generated within the VTS system by operator intervention, and are based on the last known position, course and speed of the vessel. A description of the types of sensors and the track information they generate, can be found in Chapter II.

B. THESIS OUTLINE

An introduction to the VTS environment and overall system description is provided in Chapter II. In Chapter III, data collection and the approach taken to the necessary

formatting and preprocessing needed to prepare the data for the algorithm is discussed. The discussion of the algorithm, its development and its component parts are found in Chapter IV while the actual results can be seen in Chapter V. Conclusions and suggestions for follow on work are included in Chapter VI. The Appendix contains a listing of the main fusion code and all supporting functions which were developed entirely within the MALAB¹ environment.

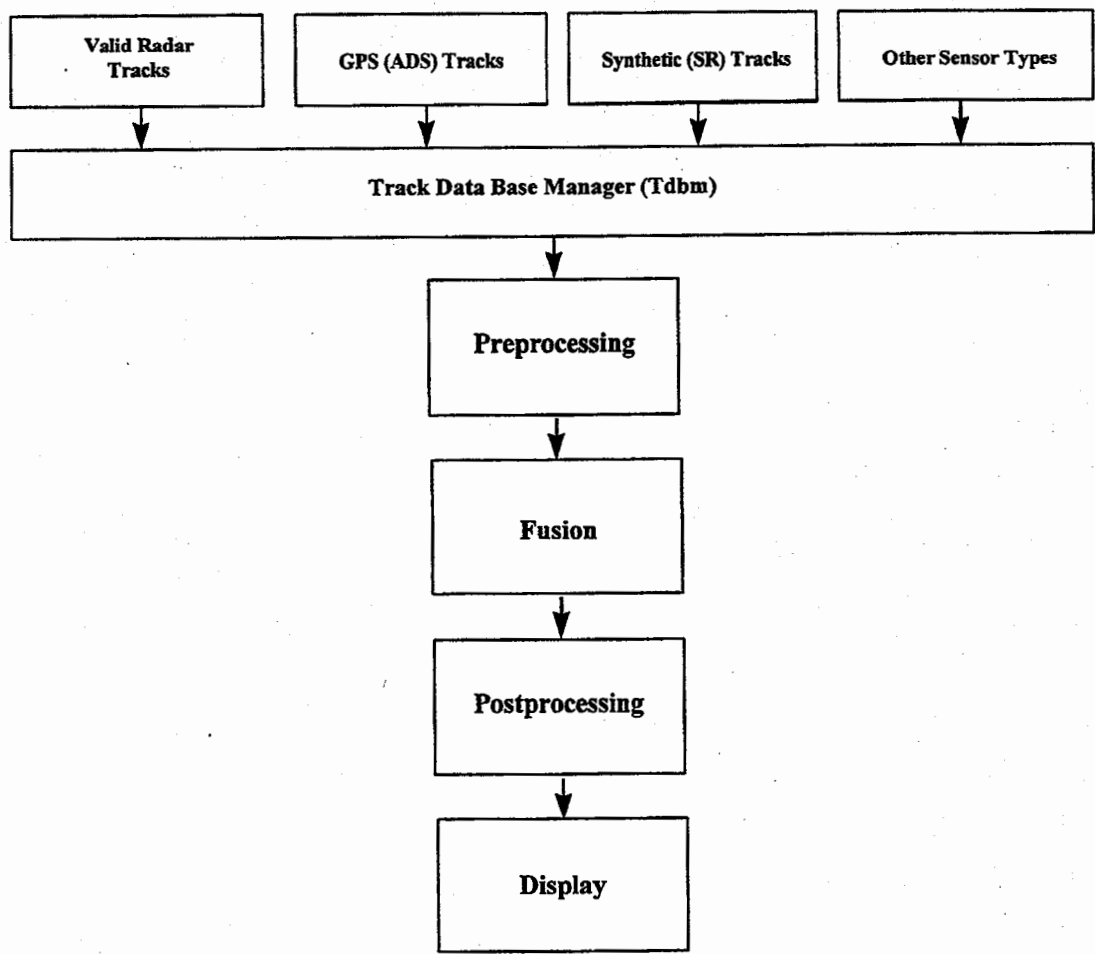


Figure 2. Overview of Fusion Algorithm

¹ A high performance numeric computation and visualization software package.

II. THE VTS ENVIRONMENT

In this chapter the salient sections of the VTS system will be presented and the actual acquisition of information, from the various sensor types, explained. The VTS system is a module of the Joint Maritime Command Information System (JMCIS). The current configuration of the VTS system is based on the Unified Build (UB) Software Development Environment (SDE) Track Database Manager (Tdbm) Service. [Ref. 2] The pivotal role the Tdbm plays in the system will be discussed and, in particular, how the various sensors provide tracks to it will be described.

The importance of or need for a data fusion scheme within the VTS system has been clearly identified [Ref. 3] for the overlapping radar coverage scenario. The problem has recently received added emphasis [Ref. 1] due to the pending implementation of the Automated Dependent Surveillance (ADS) [Ref. 4] module within the JMCIS. ADS is a Global Positioning System or Differential Global Positioning System (GPS/DGPS) vessel reporting system currently under development for integration into the VTS system. It will provide a greater reporting redundancy, thus even more uncorrelated tracks to the operator displays. A third source of information is via Estimated Positions (EPs) of vessels transiting through a Vessel Traffic Centers (VTCs) region. These are based on system defined Standard Routes (SR) which are available to the operator should the vessel in question have a non reporting status from any of the system's other sensors. The quantity of uncorrelated data continuously provided to operator displays, from these three source types, is a serious deficiency which the proposed fusion scheme will seek to address.

The VTS system is, for all practicable purposes, JMCIS with all correlation functions but Link Correlation disabled. Figure 3 [Ref. 5] is illustrative of the current VTS system configuration. In this arrangement, tracks are generated at the Remote Site Processor (RSP) and reported to the responsible VTC. At the central site, the tracks are routed through the Link Correlation module where each track is defined as a link. There is no mechanism, within the system, that will perform link-to-link correlation. Each link is

assigned an association on a one-on-one basis with a platform track. Platform tracks are sent to operator displays and are the basis for decision making. The platform tracks are also archived on a regular basis for future retrieval, should there be a need, due to an accident or for system analysis and training. Link tracks, on the other hand, remain resident in the Tdbm only until a more up to date track is reported which has the same identifying information as to RSP and track number.

As shown in Figure 3, the VTS system does not use four of the five correlation

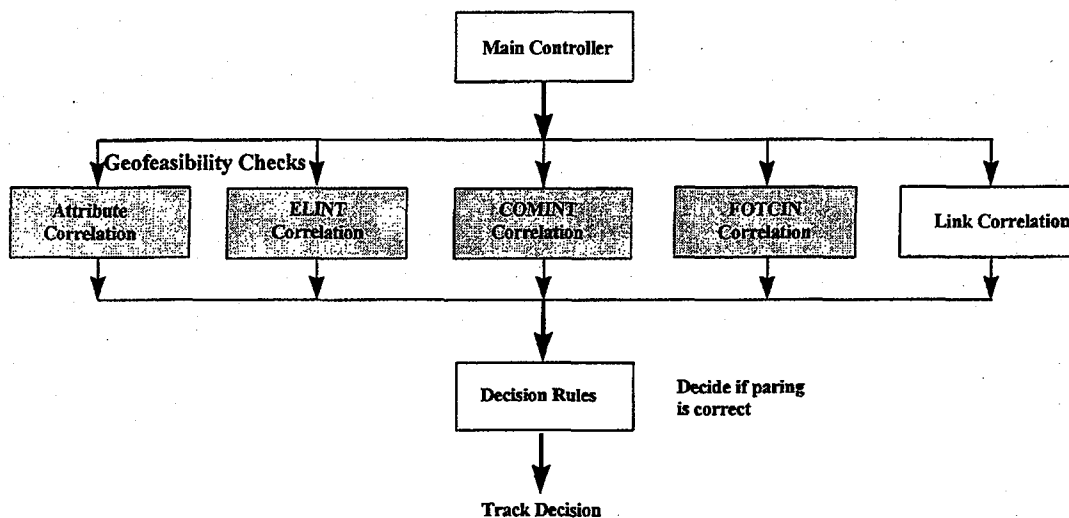


Figure 3. Current JMCIS Flow for VTS [Ref. 5]

functions of the JMCIS system. It is configured this way to ensure that the one-to-one association between a link track and a platform track is never severed. While this approach definitely accomplishes its purpose, it prevents the VTS system from being able to perform many-to-one or redundant link track associations to one platform track. This is the primary cause of multiple icons, which represent the same vessel, that tend to clutter up the display. This clutter contributes significantly to confusion and indecision amongst operators as to what is really going on in the area of concern. The fusion algorithm proposed in this thesis will make these many-to-one associations as quickly and as transparently as possible, allowing the operator to focus on overall vessel traffic management as opposed to managing multiple incidences of the same vessel.

The fusion algorithm could be introduced as a part of the Correlator as depicted in Figure 4. Figure 4 [Ref. 1] is a representation of the JMCIS software architecture. The level of detail is limited to meet the needs of this report. As tracks are reported into the Tdbm, each one is sent through the Correlator in order to promote it to an existing platform track (report from the same RSP with an identical track number) or generate a new platform track. As previously stated, all linked tracks are promoted to platform tracks which are then sent to operator displays and an archive. At this point the fusion algorithm would examine the link tracks resident within the Tdbm and determine whether any redundancy in reporting had occurred. The algorithm would then output a unique set of platform tracks where one-to-one (unique track) and many-to-one (redundant reports from multiple sensors on the same vessel) promotions had been accomplished. This would result in only the actual vessels present being displayed with little misleading information being presented to the operators. In addition, the operator would have confidence that the most appropriate information, on all vessels present, is being output to the VTS display.

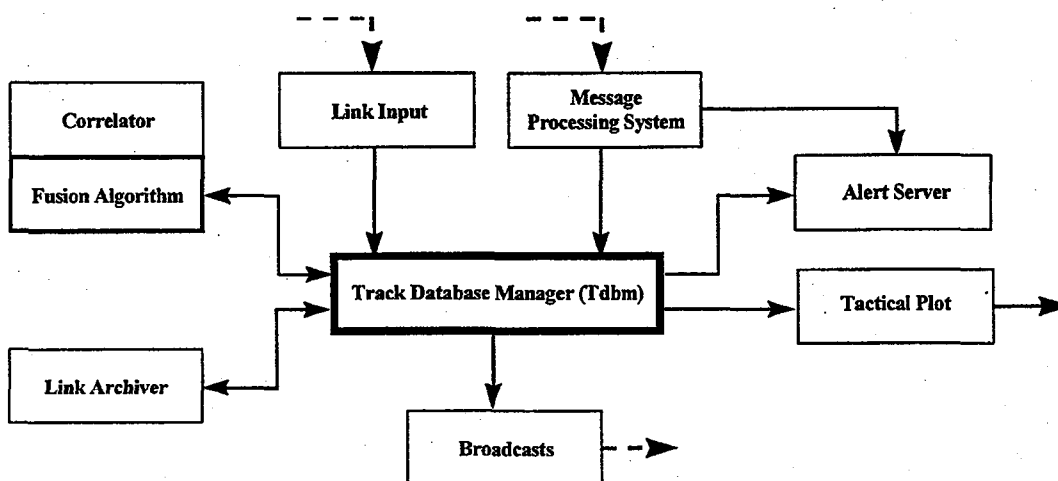


Figure 4. JMCIS Software Architecture

A. RADAR TRACKS

This section will summarize the process of how radar tracks are generated at an RSP and reported into the VTC. Detailed explanations can be found in [Ref. 1] and [Ref. 6].

The radar subsystem of VTS is currently the primary source of vessel data. The typical configuration consists of a mixture of long and short range surface search continuous wave (CW) radars with a standard set of features (STC, FTC, Sector Blanking, CFAR, Clutter Processing, etc.) and operating modes. [Ref. 1, 2, 7] In addition, the remote sites are equipped with up to four, remotely controlled, Close Circuit Television (CCTV) cameras, and the video can be called up at any display. These cameras can be color or black and white but do not have any special features for low light or night time operation.

The radar processor incorporates a sliding window detection algorithm which integrates hit data over the antenna beamwidth. It uses leading and trailing edge confidence count criteria to extract targets to achieve the CFAR (system default is $10E-5$) set by the operator. [Ref. 6] A Confidence Count (CC) is performed to determine if the required number of hits occurred to declare a valid plot. Sensor level fusion is then carried out through a pairing, developing and maturing sequence. Once a target has been declared mature, it is reported to the system at every rotation of the antenna. This continues until the target is dropped by the operator or traverses outside the sensor's coverage area. The fusion algorithm assumes that the system parameters have been optimized for the current operating (radar performance measurement and environmental) conditions and that valid tracks are being reported to the VTC's Tdbm. [Ref. 1]

The following information is sent to the Tdbm from the RSP via a microwave or fiber optic communications link: [Ref. 6]

1. Site Number (Sensor identification);
2. Track Number;
3. Time of Track Position (UTC);
4. Course in Degrees (°);

5. Speed in Knots (KTS);
6. Predicted Range in Nautical Miles (NM);
7. Predicted Azimuth in (°);
8. Radar Range in NM;
9. Radar Azimuth in (°);
10. Extent Range in NM;
11. Extent Azimuth in (°);
12. Track Quality (low of 4 to high of 9);
13. Acquisition Mode (Automatic - A, Manual - M);
14. Lost Track (Set after a predetermined number of Coast Tracks have occurred);
and
15. Coast Track (Indicates no hit on last scan).

Not all of the above information is currently used by the VTS system but is included here in order to suggest possible uses for it during future system upgrades and algorithm refinement. The quality of the data and its exact employment will be discussed in Chapter III.

B. AUTOMATED DEPENDENT SURVEILLANCE (ADS) TRACKS

The GPS based Automated Dependent Surveillance (ADS) segment is currently being integrated into the Vessel Traffic Services (VTS) System Expansion program. [Ref. 8] The ADS segment will be resident within the Data Base Processor along with the current radar and standard route modules. There are no significant changes required within the Operator Display Processor which manages the functions required for updates. These functions include the integration of raw radar video, chart generation, Tdbm events, and all alarm services (see Figure 5). ADS will be proposed as a segment of JMCIS and will use some current JMCIS capabilities. The ADS system was first tested in New York in

January 1996 and underwent further testing in Puget Sound starting in September 1996. The data for testing of the fusion algorithm was collected in September 1996 at the Puget Sound VTC. Based on the results of this effort, a program to collect more complete data sets was initiated and is ongoing.

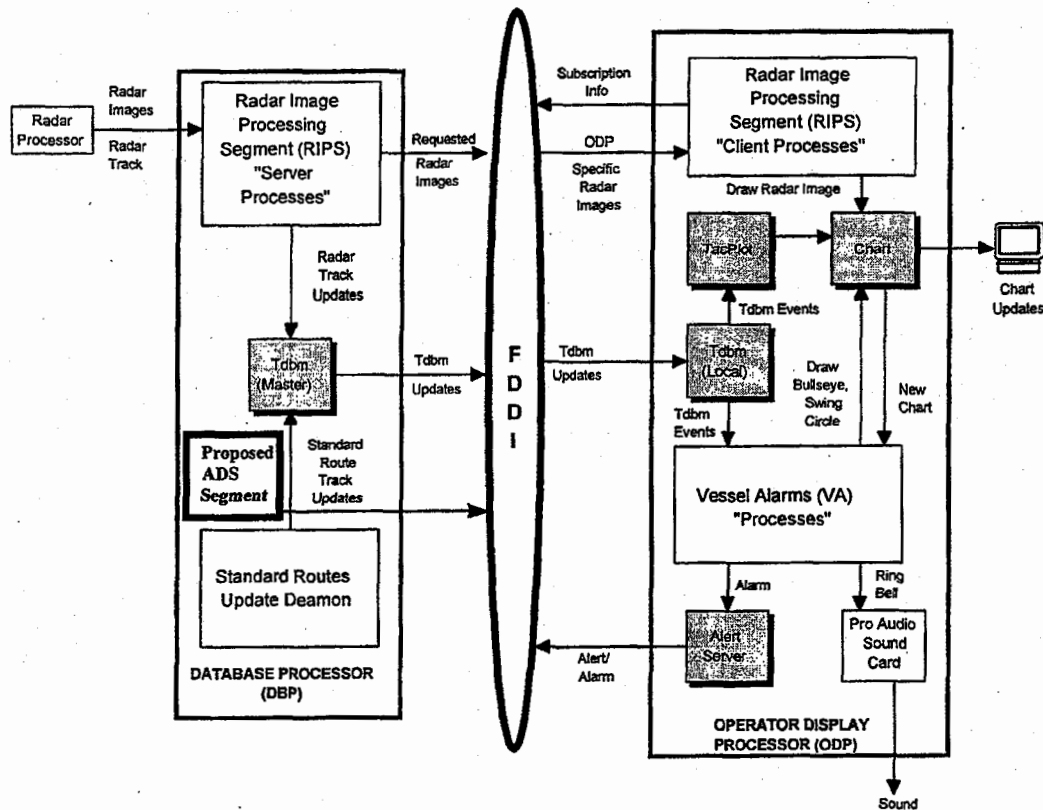


Figure 5. Proposed ADS Segment

The GPS is a U.S. Department of Defense (DOD) worldwide satellite-based navigation system, originally intended for the exclusive use of the military. GPS Standard Positioning Service (SPS) is a slightly degraded GPS signal available worldwide at no cost to any user who wants it. GPS SPS is accurate to 100 meters anywhere around the globe. Precise Positioning Service (PPS) is an encrypted GPS service used only by the military and is accurate to within 21 meters. Differential GPS (DGPS) is a USCG program to realize a 10-meter accuracy from the GPS SPS by furnishing signal corrections to properly

equipped users. This will be accomplished via existing radio beacons, and the system is currently at or near full operational capability. DGPS service covers the coastal waters of the continental U.S. as well as the Great lakes, Puerto Rico and selected areas of Hawaii, Alaska and the Mississippi River. [Ref. 7]

The ADS segment will provide GPS and DGPS tracking capability to the VTS system. The inherent accuracy of GPS based systems [Ref. 7], their relatively low cost and almost universal presence will make ADS a key component of the VTS system in the near future. ADS information is sent, from the vessel itself, to the VTC over a satellite or Digital Selective Calling (DSC) data link. The National Marine Electronics Association (NMEA) 0183 Standard is used to report ADS tracks into the system. This "Voiceless VTS" data stream provides all required information in order to build a track history within the Tdbm. [Ref. 4] The following information, on ADS tracks, is available to the fusion algorithm. (This format was used as the common data collection matrix for all sensors included in ADS trials at New York and Puget Sound.)

1. Vessel Name;
2. UTC;
3. Tracking Status (e.g., Radar, ADS, SR);
4. Track I.D. (Track Number);
5. Sensor Track Number (e.g., Radar Track Number or SR Number);
6. Course (True Course in degrees (°));
7. Speed (Knots Over the Ground);
8. Latitude (°);
9. Longitude (°);
10. Size of Vessel (Length of Vessel); and
11. Track Quality (GPS Quality Indicator or Radar Track Quality).

As with the radar data, not all the components of the data string are used but included here for completeness and their potential for future use.

C. STANDARD ROUTE (SR) TRACKS

The last source of information available to the VTS system is synthetic and generated by the system itself. These tracks are referred to as Standard Route (SR) tracks and represent an Estimated Position (EP) of the vessel of interest through a VTC's Area of Responsibility (AOR). SRs are generated by the SR daemon, and the system can be configured for automatic or manual generation.

Typically, SRs are generated when a radar track is lost on a vessel. This can occur for a variety of reasons including radar blind spots, atmospheric conditions and or a malfunction of the sensor itself. Once this occurs, an SR is initiated to help estimate the position of the vessel as it transits the AOR. These SRs are multisegmented predefined routes that are geographically fixed to represent the waterway under consideration. These routes are assigned based on the type of vessel and initial position, and vectoring is derived from the track information last reported into the Tdbm. The predicted path of the vessel is then updated every ten seconds into the Tdbm and closely monitored and manually updated as deemed necessary. The SR is terminated once the original or another sensor acquires the track. This process is operator intensive and not well understood. There is no association between radar and SR tracks and it takes a great deal of operator experience and intuition to generate a reasonable approximation of a vessel's route. [Ref. 1]

Reference 1 describes two common scenarios where SRs are used. A change to the way SRs are currently implemented is suggested that would provide for continuity of platform numbers as vessels are handed off from a sensor to a SR then to another sensor. The proposed fusion algorithm performs this task, requiring only that the vessel parameters assigned within the SR generation process be a reasonable representation of actual track parameters. This will ensure that it can be fused to a sensor generated track once it is detected again.

III. DATA COLLECTION, FORMATTING AND PREPROCESSING

This chapter covers the details of data collection, the formatting used and preprocessing required to prepare it for input to the fusion algorithm.

A. DATA COLLECTION

The data used to test the algorithm was collected over a two day period, September 11-12, 1996, at the Puget Sound VTC (see Figure 6). Data collection was piggy-backed on the efforts of the prime contractor, Inter-National Research Institute (INRI) and the USCG who were conducting ADS trials. Conditions for the data collection were satisfactory, and data sets were rich with multiple sensors, primarily radar and ADS, reporting on the same target. There were no equipment malfunctions that biased the data and no particular "tweaking" of sensors or other system components took place. Weather conditions were conducive for radar operations, and there were no communications deficiencies. Portable ADS equipment was set up on selected Washington state ferries whose routes and schedules were well known to the VTS system operators.

Track history recording was conducted in accordance with [Ref. 9] and on a non-interference basis with normal VTC operations. Up to 10 tracks were available for simultaneous track history recording for up to 12 hours in duration. This equates to a file size of approximately 15 MB (maximum).

Track quality (TQ) and size of vessel parameters were not available and defaulted to zero by the collection team. The issues surrounding track quality and vessel size will be addressed in a later section. All other parameters were correctly reported into the system and archived.

The methodology used to select track data was highly discretionary but effective. Operators would examine their displays in order to determine if any interesting scenarios were developing. Great care was taken to ensure that ADS and radar tracks that could even

SEATTLE TRAFFIC FREQUENCY AREAS AND REPORTING POINTS

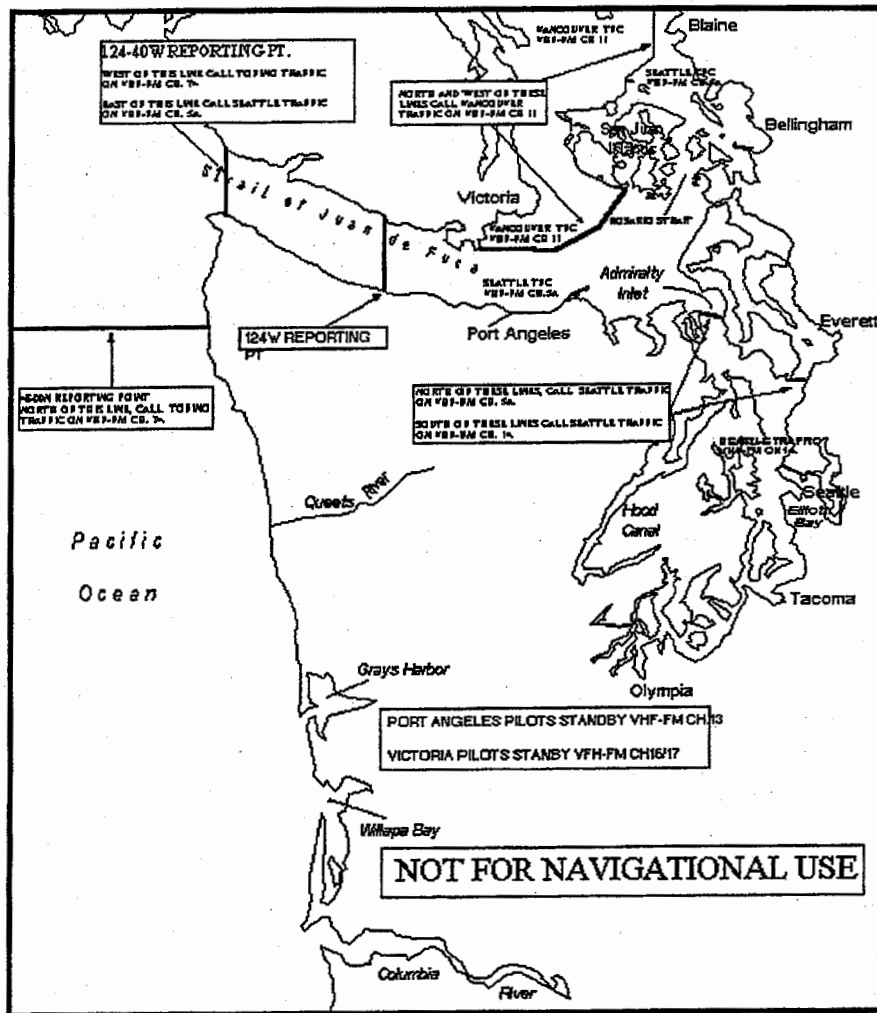


Figure 6. Test Area

remotely be of the same vessel were recorded until the situation resolved itself or a more interesting scenario started to develop. Post analysis of the data showed that there was enough variety in the scenarios and sufficient redundancy in sensor reports to thoroughly test the fusion algorithm. Due to the limited number of tracks that could be recorded, emphasis was place on ADS and radar data; no SR track scenarios were collected. Due to the limited number of scenarios that could be collected at one time, individual scenarios

were not collected in their entirety but dropped once they became less useful than other situations that were starting to develop. These shortcomings need to be addressed during future data collection exercises.

B. FORMATTING

The track histories were stored in ASCII files and the data was recorded for each selected track in the following format [Ref. 8]:

Name: track-history.dat										
Path: /h/data/local/ADS										
Format:										
1	2	3	4	5	6	7	8	9	10	11
cccc	DDMMY Yhmmss	AAA	cccc	cccc	x.x	x.x	ddmm.mm	ddmm.mm	size	x<CR>
cccc	DDMMY Yhmmss	AAA	cccc	cccc	x.x	x.x	ddmm.mm	ddmm.mm	size	x<CR>
etc.										
1	26cc	Vessel Name								
2	DDMMY Yhmmss	UTC—Time of Track Position								
3	AAA	Track Status (Radar, SR, ADS)								
4	cccc	Track ID								
5	cccc	Sensor Track #—Radar # and Track # or CID # or SR #								
6	x.x	True Course								
7	x.x	Speed (knots over ground)								
8	ddmm.mm	Latitude—degrees minutes								
9	ddmm.mm	Longitude—degrees minutes								
10	#	Size of Vessel								
11	x	Track Quality (good, coast, lost for Radar; 0, Non-Diff, Diff for ADS)								

A sample of the contents of a recorded track file is shown below:

```

UNK-4743,110996212055,Radar,742,3,180.4,5.9,4735.08,-12228.05,0,0
UNK-4754,110996212055,Radar,753,3,186.6,5.1,4735.54,-12227.82,0,0
UNK-4773,110996212052,Radar,772,3,93.4,18.0,4736.41,-12228.42,0,0
SPOKANE_ADS,110996212023,ADS,773,3669994520,92.7,18.0,4736.37,-12228.66,0,0
SPOKANE_ADS,110996212056,ADS,773,3669994520,93.2,18.3,4736.36,-12228.41,0,0
SPOKANE_ADS,110996212056,ADS,773,3669994520,93.2,18.3,4736.36,-12228.41,0,0
UNK-4751,110996212058,Radar,750,3,357.7,8.9,4738.47,-12226.49,0,0
UNK-4756,110996212058,Radar,755,3,195.2,9.2,4734.51,-12228.03,0,0
UNK-4773,110996212058,Radar,772,3,91.9,18.1,4736.41,-12228.38,0,0

```

A vessel's name is identified in column one as unknown (UNK-XXXX) if the true identity has not been determined. Column three indicates sensor type. All other columns are self explanatory.

C. PREPROCESSING

Due to the straight forward format used to collect the system data, almost no preprocessing of individual sensor data was required (for formatting purposes) as was the case in [Ref. 1]. The fusion algorithm simply reads the data files to extract the relevant track parameters. The data files were recorded in ASCII format with each field delimited by a comma. The following procedure was then utilized to build data files:

1. Open up the ASCII file in the word processor of choice [Microsoft Word® in this case];
2. Cut and paste the desired length of data into a new file, then use the Save As menu choice and name the file with a *.dat* extension. [e.g. *11_e1.dat*]; and
3. Perform a global search and replace on Track Status, changing Radar to a "1," ADS to a "2" and SR to a "3."

Once these files were built, a scheme was required to allow MATLAB® to read the various fields and emulate what the data would look like to the fusion algorithm within the Tdbm. To accomplish this, a function was written called *getdatax.m* (see Appendix) which reads the data line-by-line and places it in a matrix called *ObsnMatrix*. Within *getdatax.m*, descriptive variable names are assigned that mirror the actual data fields; the data fields are padded, where appropriate, to accommodate fields that have variable lengths. The data from the individual fields are placed in their respective storage vectors. Once all of the data has been read, they are appended together to form the observation matrix *ObsnMatrix*. The contents of the observation matrix are easily discerned via their descriptive names, and the order of the data fields has been changed to the following:

```
ObsnMatrix = [ Latitude  
               Longitude  
               TrueCourse  
               Speed  
               Size  
               TrackIDNumber  
               UTC  
               TrackQuality
```

TrackStatus
SensorTrackNumber]

No part of the information from the original data set is discarded. It is simply placed in a storage vector and available for recall should there be a need for it. A good example of this is *VesselName* which was deliberately excluded from the observation matrix because it was redundant (*TrackIDNumber* provides the same information) and subject to human error via misidentification of the vessel in question.

The attractiveness of this data capture and preprocessing scheme lies in its simplicity, robustness and adaptability to accommodate any type of data file arranged in a similar fashion. It is merely a matter of knowing how the fields are delimited from each other and then determining the maximum length of each field. Even this process could be automated, within the code, making the routine that much more multifunctional. The output depends only on what the user needs and can be arranged in any order that is desired.

The data is now ready to be fed into the fusion algorithm as if it were available in real time, as would be the case in the actual VTS System. Windowing of the data and the selection of tracks chosen for processing within the fusion algorithm will be described in the Chapter IV as they are integral to the performance of the fusion algorithm.

IV. FUSION ALGORITHM

This chapter examines the fusion algorithm in detail (see Figure 7) and describes the fuzzy association techniques that are employed to provide a possible solution to the growing track redundancy problem within the VTS System. The chapter starts with how the data are windowed prior to the fusion algorithm. Multisensor data fusion, in general terms, is discussed and then related to the approach taken within the fusion algorithm.

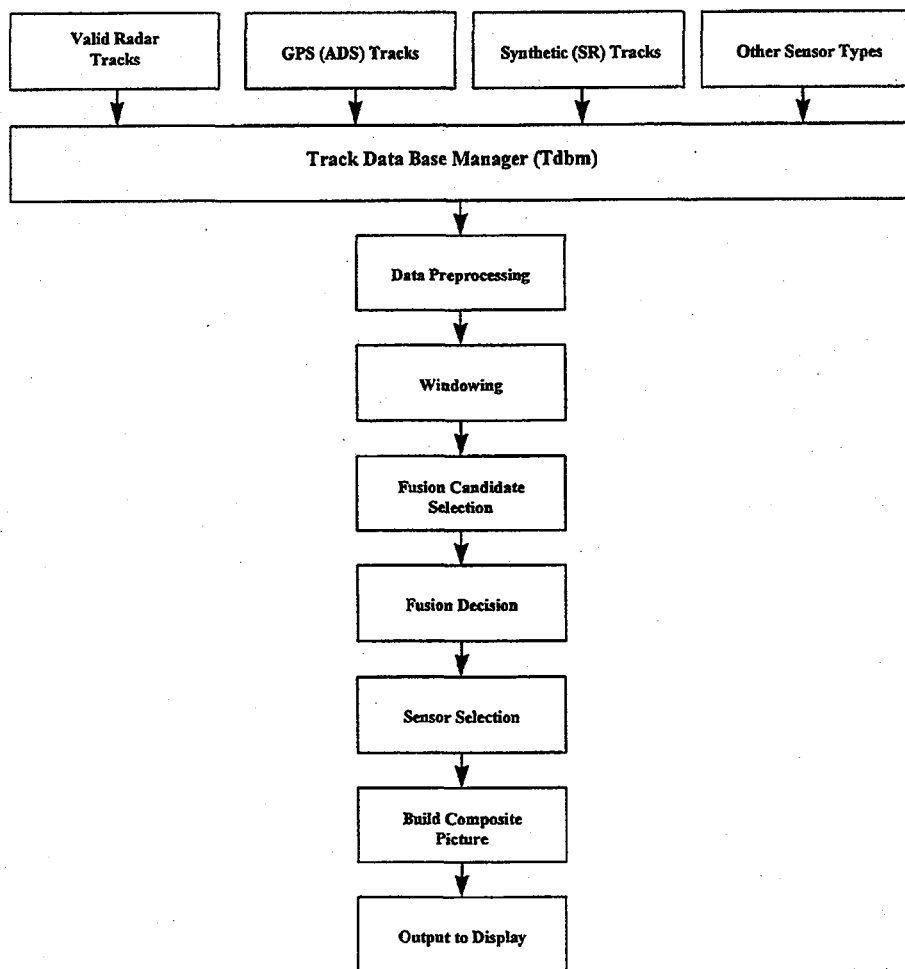


Figure 7. Overview of Fusion Algorithm

A. WINDOWING OF THE AVAILABLE DATA

A time window operation is applied to the data once it is made available to the algorithm from the Tdbm (utilizing *getdatax.m*). At this point, there exists a requirement to eliminate the redundant track reports, thus reducing the data set that resides in the time frame to be considered for analysis. This is accomplished by applying a time window to the data set. MATLAB® function *timWin.m* (see Appendix) performs the window operation within the main program *fusex.m* (see Appendix). The windowed observations are placed in a refined observation matrix called *WindowedObsns* to await possible further data set reduction.

The actual length of the window depends strictly on update rates from the various sensors and the relative change in position of a track between updates. In the case of the VTS System, radar tracks are updated every six seconds, SR tracks every ten seconds and ADS tracks every 15 seconds. Vessel speeds vary from zero to twenty knots with the majority of vessels making good around eight knots. The faster vessels tend to be the local ferry traffic.

Given the relatively slow change of position of the vessels and the fast update rates of the sensors, it is readily apparent that the tracks are over sampled for this application. A window size of 15 seconds would ensure an opportunity for all sensor types to report thus addressing any latency issues for the current system configuration. The VTS System does not require that positions be updated this quickly; therefore, it is possible to substantially reduce the processing load by optimizing the window size to obtain a satisfactory update rate. This process however, still leaves multiple reports, on a track, from the same sensor within this window which will need to be addressed. The key is to make sure that the most recent report from each sensor is resident in the time window that is utilized for analysis.

After the time window has been applied, the algorithm then selects the most recent track from the *WindowedObsns* selected for analysis with a call to the function *mstReTr4.m* (see Appendix). This function selects the most recent tracks assigned to unique track numbers and places them in a further reduced observation matrix called *MostRecentTrks*.

The data set is now reduced to the desired content and can be passed to the fusion algorithm for processing.

As with the data acquisition algorithm, *getdatax.m*, *timWin.m* and *mstReTr4.m* (see Appendix) are simple, robust, adaptable and generally useful for a wide range of applications that use similar types of data sets. Window lengths can be determined by an examination of the latency associated with different sensor types and the relative change in position of a target between updates.

B. MULTISENSOR DATA FUSION

The primary goal of the fusion algorithm is to fuse together observations from different sensors made on the same target or vessel. The reporting sensor can be of any type as long as it provides the necessary information upon which fusion decisions are made. In this thesis, reports are available from radar, ADS and SR mechanisms as described in Chapter II. The inclusion of acoustic sensors positioned in critical waterways, or in fact any other type of sensor, can be readily achieved in future versions of the algorithm. The fused tracks are assigned a platform number for output to the operator displays, with the superior sensor assigned reporting responsibility. The information from inferior sensors is suppressed, but not decimated, resulting in a more clear picture of what is occurring in the waterways and harbors via operator displays. This fusion process is achieved through the use of fuzzy membership functions as an approach to determining a level of correlation between the set of observations from different sensors.

1. Fuzzy Association

The proposed algorithm is an automated reasoning facilitator which attempts to capture the inference capability of VTS system operators (or experts as they are usually referred to in the literature on fuzzy logic). It attempts to accomplish this by specifying rules that help in making decisions or illuminate the essential elements of a task. [Ref. 10] Knowledge Based Systems (KBS) attempt to segment the problem into sub-problems and then, using the rules established by the expert, solve the problem interactively. The overall

problem is finally solved by combining the results of the sub-problems. In this case we associate redundant tracks for same vessel which are reported into the system from the various system sensors. This association is performed by the membership functions that are used to measure the correlation, or level of similarity, between a set of observations. These values of "sameness" are then used in the fusion process for decision making (threshold setting) and track identification. The output can be visualized as the result of taking a combination of data, from different sources, to obtain a refined location and identity estimation on the target. [Ref. 11] In essence we have improved our knowledge of a target's location by taking advantage(via association techniques) of the reporting redundancy inherent in the VTS System.

2. Levels Of Fusion

Three levels of fusion are generally considered in the literature. Level 1 or Positional fusion is the lowest level in the data fusion hierarchy and seeks an improved estimate of target position using parametric data. [Ref. 12] Level 2 is Identity Fusion which uses a target's characteristics to determine its identity while Level 3 is called Threat Assessment and used in military applications to fuse data in order to determine an appropriate response to an adversary. [Ref. 11, 12, 10] Positional fusion is the appropriate level of data fusion for the VTS shortcomings that this project is trying to address. The manner in which Level 1 fusion techniques are applied to solve these shortcomings will be the focus of the rest of the chapter.

C. POSITIONAL FUSION

1. Sensor Level

The reporting sensors in the VTS System have already performed positional or sensor level fusion prior to initiating a report to the Tdbm. In the case of radar, it is a function of the pairing, developing and maturing sequence referred to in Chapter II. In the case of ADS and SR reports, it is physically impossible to have overlapping coverage on a vessel. The VTS system assumes that sensor level fusion is being carried out correctly and

only valid, non-redundant tracks are being generated and reported by individual sensors.
[Ref. 1]

2. Central Level

Once these valid tracks are in the Tdbm, central level positional fusion is carried out to eliminate the redundancies that occur from different sensors reporting on the same track. This is not database fusion as the algorithm does not destroy or alter any information about a target even though a fusion decision may have been made. The output from the inferior redundant sensors is simply suppressed and not routed to the displays. This approach was taken to ensure that the system could take advantage of track redundancy as represented by the suppressed information. This suppressed information would be utilized if the reporting sensor on a fused track ceases to report and a hand off to the next superior sensor becomes necessary. The other obvious case is when a decision is made to defuse. Additionally, having this information available for ready recall helps in the analysis of the system to ensure optimum performance.

D. FUSION PROCESS

The algorithm (see Figure 8) now takes the reduced data set resident in the matrix *MostRecentTrks* and begins the fusion process. It accomplishes this by sequentially comparing track pairs in order to determine the grade of membership between the attributes that are used in the fusion decision process. The attributes used in the decision process are *Latitude*, *Longitude*, *Course* and *Speed*. Originally, vessel *Size* and *TrackQuality* were to be included but their utility was marginalized by the methods used to record their values into the system during data collection. If deemed necessary, these or any other suitable attributes can be readily added in the future due to the modular approach used to construct the code.

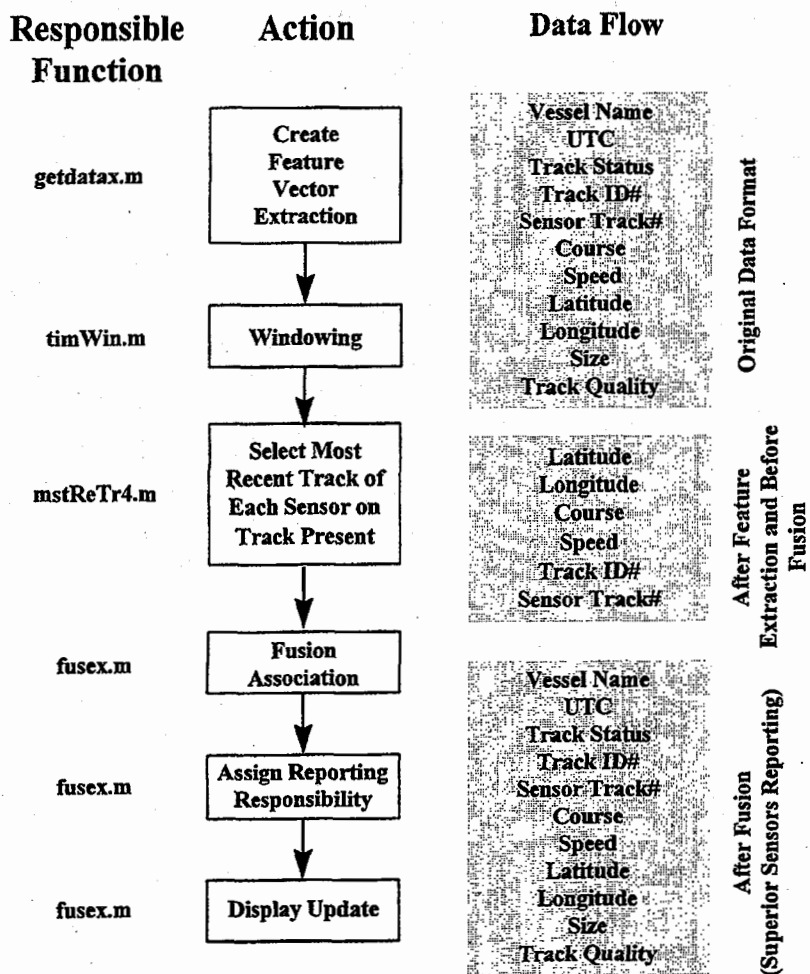


Figure 8. Flow Chart of the Fusion Algorithm

1. Membership Function Design

The assignment of membership value that is accomplished by the fuzzy association system is a measure of similarity or sameness by correlation. [Ref. 1] Fuzzy Set theory as opposed to traditional set theory, considers the partial relationship or membership of an object in a set. Membership functions are used to grade the attributes of a set usually in the range [0,1]. The closer the attribute is graded to the upper bound the higher the grade of membership. The higher the grade the attribute is assigned, the more similar it is to the attribute it is being compared against. Instead of answering the question with a crisp or

simple YES or NO, it provides a scaled interpretive answer which can be NOT LIKE, A BIT LIKE, SOMEWHAT LIKE, A LOT LIKE, or LIKE. This type of answer is obviously a better representation of how VTS operators currently interpret about what is developing on their displays.

In the design of a fuzzy association system the following approach is used: [Ref. 2, 13]

- Ascertain the universe of discourse of system input(s) and output(s).
- Design the membership function(s).
- Decide on the fuzzy rule(s) to relate input(s) and output(s).
- Devise the defuzzifying technique(s).

Membership function design is based on the variations inherent in an attribute that is being compared. [Ref. 1] Given that radar and ADS positional reports (i.e. latitude and longitude) are for the most part dependable and accurate, a form of triangular membership function is often used as shown in Figure 9. Where as attributes that tend to be not quite as accurate (highly dependent on the type of sensor), such as speed, require a broadened roof as shown in Figure 9, which allows for a greater range of values. Combinations of these typical membership function shapes are useful, as in the case of the course attribute, where you desire a generous association within a certain range but not outside of a fixed range. This gives a trapezoidal shape to the membership function as can be seen in Figure 9. Membership functions are by their very nature subjective, but they are far from arbitrary and need to be based on the application and the attribute in question. The relative shape of a membership function is only a starting point and follow up analysis of its performance is critical to fine tuning the process. A more in depth discussion of membership function design can be found in [Ref. 1, 2, 13].

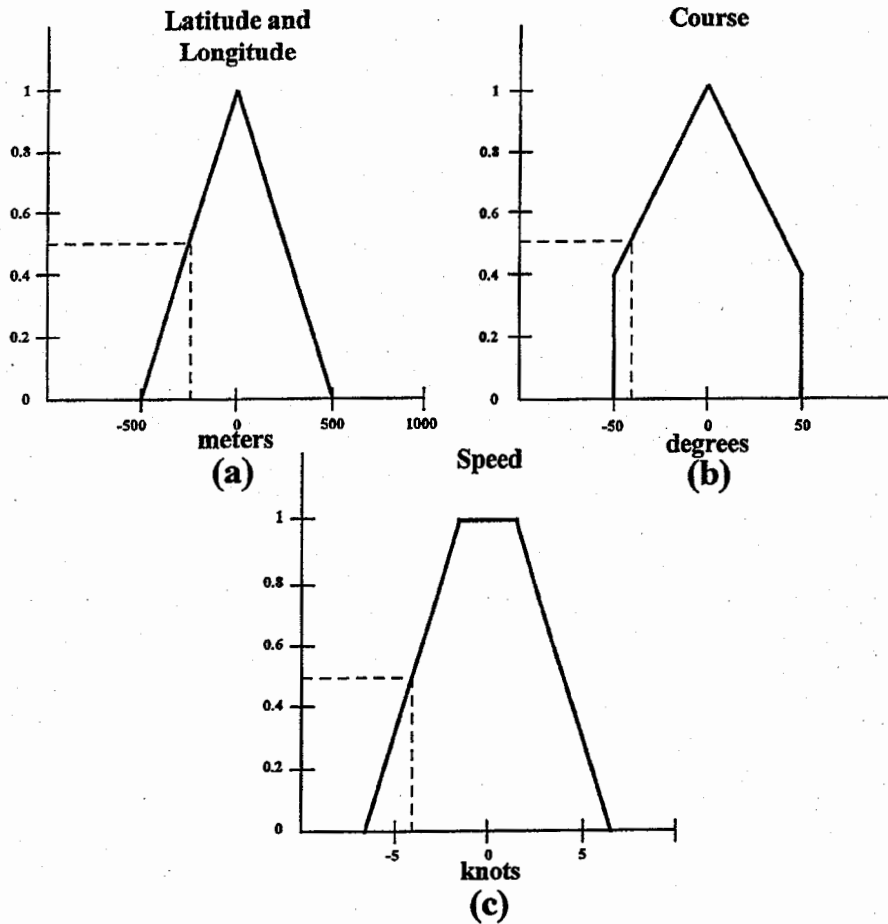


Figure 9. Membership Functions: (a) positional attributes - longitude and latitude; (b) course in degrees; and (c) speed in knots

2. Threshold Test

The next step in the process is to evaluate all the attributes and their membership grade against a threshold value. This threshold value represents the known physical limitations or specifications of the sensor. In the case of a radar, it is based on bearing resolution, range resolution and speed error. For ADS, it is the relative accuracy of the measurements based on the type of the GPS being used. With this diversity in the relative accuracy of sensor attributes, the threshold is always set based on your least accurate sensor.

3. Fusion Decision

With the thresholds set, the membership values of the attributes are sequentially checked in the following sequence: *Latitude*, *Longitude*, *Course* and *Speed*. Each attribute's membership value must exceed the threshold or the association fails for that track pair, and the algorithm proceeds to the next track pair and repeats the process. The GO/NO-GO decision at each attribute level reduces the run time of the algorithm contributing significantly to the overall computational efficiency of the program. Track pair combinations that have all their attributes exceeding the threshold values are defuzzified and output as a virtual binary '1' as represented by their presence in a storage matrix called *FusionCandidates* (see Figure 10).

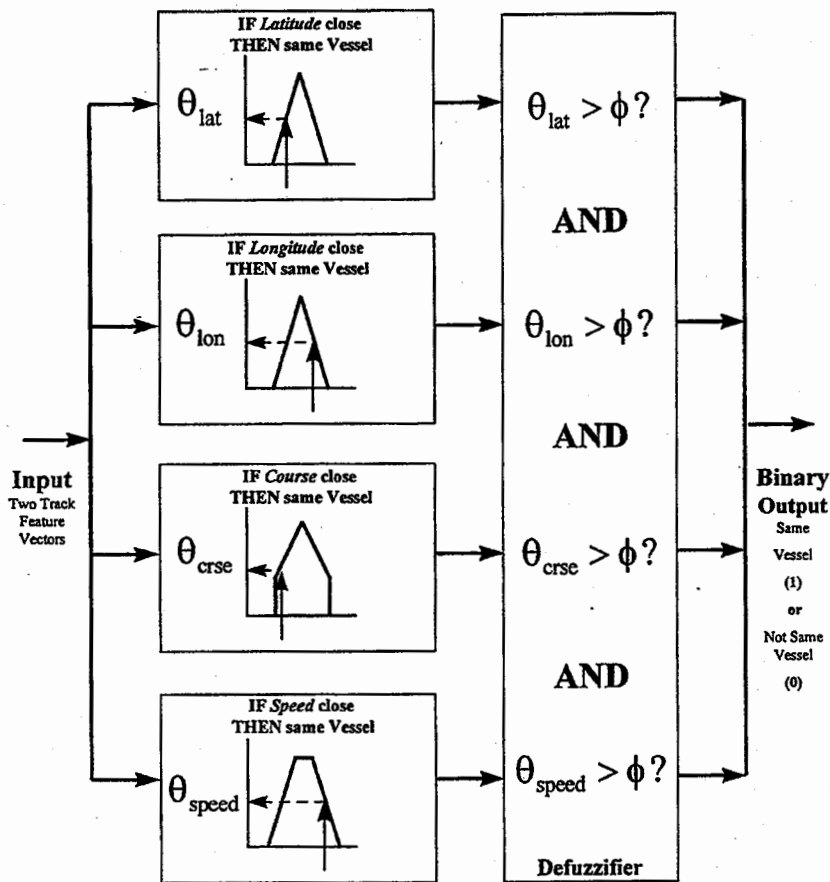


Figure 10. Depiction of the Fuzzy Associative Decision System

4. Reporting Responsibility

Once the *FusionCandidates* matrix is complete the algorithm then performs an evaluation to determine what type of sensor is reporting and its location. This information is used to assign reporting responsibility to the superior sensor. The current hierarchy has radar at the top followed by ADS and SR in order of descending priority. Radar is currently given superior sensor status due to the slow update rate of the ADS tracks. Once the update rates for ADS are at least comparable to radar update rates, ADS tracks will be assigned superior sensor status. [Ref. 14] This change is anticipated with the actual integration of ADS into the VTSS.

Should the redundancy in reporting be a consequence of the same type of sensor, it is necessary to select the superior of the two. In the case of radar tracks this is based on the characteristics of the radar; the radar possessing superior characteristics (resolution capabilities) is chosen. If the radars are similar, a designation within the system based on alternate criteria, such as current operating performance and relative distance to the target, would be used to select the superior sensor. The fusion algorithm is easily modified to accommodate any changes to sensor status. Same-type sensor redundancy is a radar issue exclusively as it is physically impossible to get multiple ADS and SR tracks on the same vessel. At this point the selected tracks that are being reported on by the superior sensors are placed in a matrix called *HitsToKeep*, and the tracks deemed redundant are placed in a matrix called *CeaseReport*.

5. Report Generation and Output

At this point it is necessary to include the tracks that were previously deemed not fuseable along with those that have been given reporting responsibility for the fused tracks. *HitsToKeep* is augmented with these lone tracks, and the matrix *UpdateReport* contains all the track numbers that need to be reported to the operator display.

The last step that needs to occur before updating the display is to take the track numbers from *UpdateReport* and extract all the track data from *MostRecentTrks* required for a complete report. For computational efficiency, all unnecessary data fields had been

purged during the fusion process. *UpdateReport* is then checked for redundancy, sorted by track number and placed in the final output matrix *TrksToPlot*.

The information is displayed on a plot (longitude and latitude on the vertical and horizontal axis, respectively). The axis limits are set by the minimum and maximum values present in the data. An example is depicted in Figure 11. Each track is assigned a permanent color the first time it appears in *TrksToPlot*. Every time a window of data is output to the plot after the fusion analysis is complete. For large over-sampled data sets

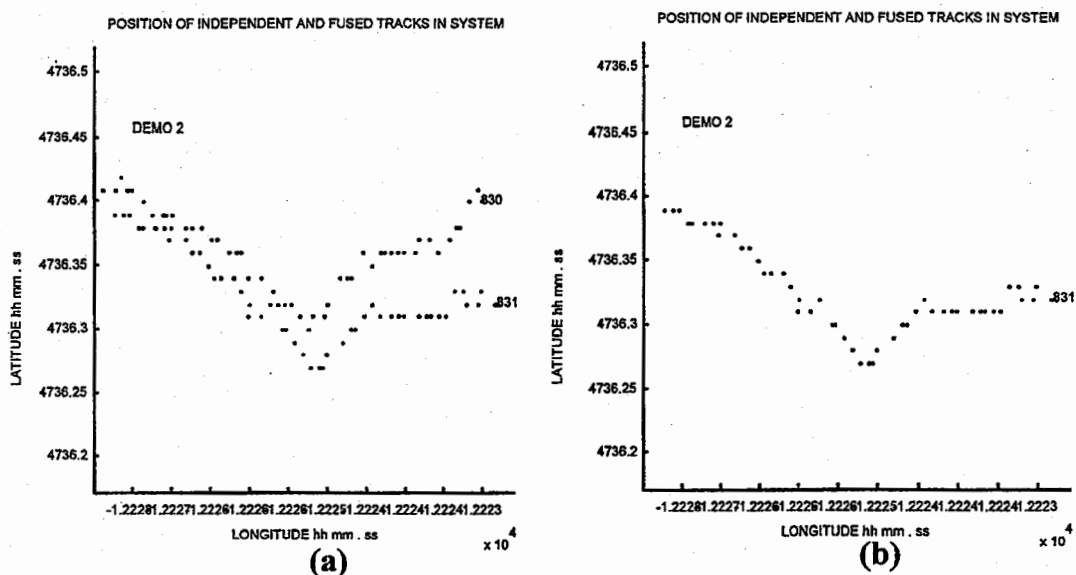


Figure 11. Scenario Plot Example: (a) no fusion applied and (b) fusion applied

with many tracks, it is possible to speed up the algorithm for quick play back by increasing the value of *windowInc* in *fusex.m*. This ability to replay the scenarios in a timely manner is very useful when analyzing the algorithm and the overall system performance.

At this point the fusion cycle is complete. The superior sensors have been assigned reporting responsibility for tracks that had redundant or multiple sensor reports. The reports deemed redundant have had their output suppressed, and the system operator is now seeing only single realizations of vessel tracks. The data *window* is now moved forward in time in

order to process the next set of sensor reports on tracks present in the system. The fusion operation is repeated in this manner until physically turned off.

The fusion process runs in the background and is capable of handling as many tracks as there are colors in the matrix *ColorOrder* in *fusex.m*. The absence of data in any given time window does not cause the algorithm to balk as it is designed keep sliding the data window until it finds track information. This is useful when there are system level problems that cause data blackouts for short or long periods of time. Until new data is available the display will retain all the information received up to the point where the data stream was interrupted. This allows for quick analysis of the situation by system operators once the data stream has been restored.

F. SUMMARY

In this chapter the fusion algorithm and the principles upon which it was designed were described in detail. Data set reduction, levels of fusion used and the membership function design were thoroughly covered. The process is an automated KBS scheme that improves our knowledge of a target's location by taking advantage of redundancies inherent in the VTS system. The end result is a clear, composite picture of what is actually occurring in the harbors and waterways being controlled by the system operators. The next chapter describes the results of this effort as applied to several real world situations.

V. RESULTS

Actual data from an operational VTS system was collected at Puget Sound in September 1996. This fortuitous availability of "real world" information negated the requirement to simulate data as had been the case in the previous work. [Ref. 1] This data allowed for thorough testing of the algorithm for a variety of real life scenarios which were chosen purposely by USCG and ADS contractor personnel. These scenarios realistically depict the redundancy issues faced by system operators with overlapping information from multiple radar and ADS tracks. These demonstrations clearly show how well the algorithm works to address this problem.

A. DEMONSTRATION CONSTRUCTION

In order to build the demonstration data sets, it was necessary to load a large amount of data, via the main algorithm, with the fusion process turned off. The output to the display is a true realization of what was occurring in the harbor and waterways during that time period. The display was then examined to determine the track numbers that were to be extracted to build a given demonstration of a particular scenario.

Within the MATLAB® command window all variables in the workspace except for *ObsnMatrix* are cleared. Column six of the *ObsnMatrix* is then examined via the *find* command to ensure the tracks of interest are present. Once this is done, a series of row vectors is created which contain the row indices of the track data of interest. The following steps demonstrate how a three ship scenario, track numbers 772, 773 and 774, is built once MATLAB® is loaded with the correct path set:

- enter *fusex*
- when prompted, enter "0" on all button bars representing fusion criteria
- when prompted, enter the file name (e.g. 11_e0.dat)

- examine the display for tracks of interest and note track number
- clear all variables in the work space except ObsnMatrix
- enter `a = ObsnMatrix(:,6)`
- enter `aa = find(a == 772)`
- enter `aa = aa'`
- enter `bb = find(a == 773)`
- enter `bb = bb'`
- enter `cc = find(a == 774)`
- enter `cc = cc'`
- enter `dd = [aa bb cc]`
- enter `new = ObsnMatrix(dd,:)`
- enter `ObsnMatrix = new`
- clear all variables in the work space except ObsnMatrix
- save demoX (saved as **demoX.mat** in current directory)

Once this procedure has been completed, the demonstration file is ready to run. The following procedure is used to run the demonstration:

- turn off the data acquisition call `getdatax` by placing a % in front of it within `fusex.m`
- enter `load demoX`
- enter desired fusion threshold criteria on button bars
- examine results on the plot

This simple procedure greatly reduced the time spent in loading the large data files and allowed for timely repetitive testing of the algorithm for the various scenarios. Demonstration files are easily modified in order to examine time frames of particular interest. The *zoom* command within MATLAB® was very useful in analyzing results in a particular region.

There are many variables within the algorithm that can be displayed during execution that will help determine what is actually happening. The Appendix contains a list of the more useful variables and their functionality. These variables can be simply output to the screen and observed as the plot develops. This will slow execution time down but this is necessary to allow the observer to keep up with what is occurring on the display. The comments within the code should allow for easy identification of critical parameters and variables. The variable names themselves are not cryptic and are standard terms used in the literature on the VTS environment.

B. RESULTS

The algorithm performed correctly under all test scenarios. The redundant tracks would stay fused as long as each track pair being assessed had a data point within the observation window. There were no problems associated with vessels that were turning and the algorithm always selected the superior sensor. The algorithm had no trouble dealing with a large amount of tracks and or interruptions in data streams. The following are representative of the types of situation the USCG would like resolved within the VTS:

- Scenario 1: Overlapping radar coverage (Tracks 830 and 831) on a single vessel. Track 831 is the superior sensor. See Figure 11.
- Scenario 2: Overlapping radar coverage (Tracks 750 and 751) on a single vessel along with an independent vessel (757). Track 751 is initially the superior sensor but drops track causing reporting responsibility to be handed off to track 750. See Figure 12.

- Scenario 3: Overlapping radar coverage (772 and 774) and ADS coverage (Track 773) on a single vessel. Track 773 is the first to acquire the vessel but hands it off to track 772, once 772 acquires the track due to its superior status. Track 774 then acquires track and takes a hand off from 772 due to 774's superior status. See Figure 13.
- Scenario 4: Overlapping radar and ADS coverage on multiple tracks over an extended period. This demonstrates the algorithm's ability to handle many tracks and the potential for a much less cluttered display. See Figure 14.

Many other scenarios were examined and the algorithm performed well in all circumstances. In summary:

- The algorithm fused all tracks that were in the overlap region that met the fusion criteria.
- The algorithm would change reporting responsibility for a track to the next inferior sensor if the superior sensor ceased reporting.
- The algorithm would change reporting responsibility for a track to a more superior sensor if that sensor started to report on a vessel which was currently assigned to a less capable sensor.
- The algorithm had no trouble with crossing or passing situations.
- Marginal situations were easy to discriminate as the algorithm would defuse immediately upon failure of the fusion criteria.

The key observations to be made are the affects that the individual membership functions had on the results. If the membership function was not sufficiently broad enough the decision to fuse two tracks was not made. This is particularly true for the course membership function. Vessels that are going extremely slow and or turning tend to have widely varying headings from the radar reports. The addition of fusion parameters, such as size and track quality, would certainly provide a greater degree of confidence in situations where position, course and speed are very close. While the data collected did not contain this type of situation, it is reasonable to assume that this scenario is common in the busy

harbors and waterways under the USCG management . These findings are consistent with the simulated overlapping radar results reported in [Ref. 1].

In summary, the algorithm fused redundant tracks and output the superior reporting sensors data, on that track, to the display while suppressing the inferior sensor reports. The algorithm was able to handle redundant reports from all sensor types represented in the data. The resulting clarification of the situation in the harbors and waterways should help to alleviate the current redundant track problems common on the VTS displays.

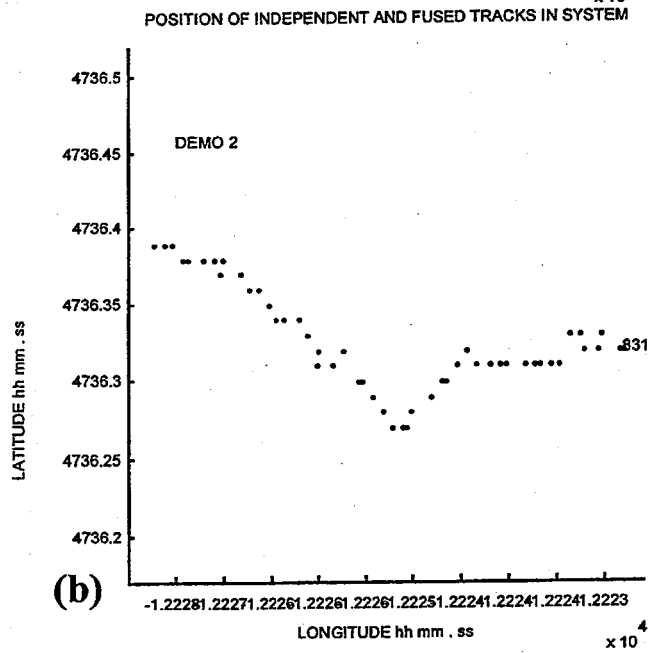
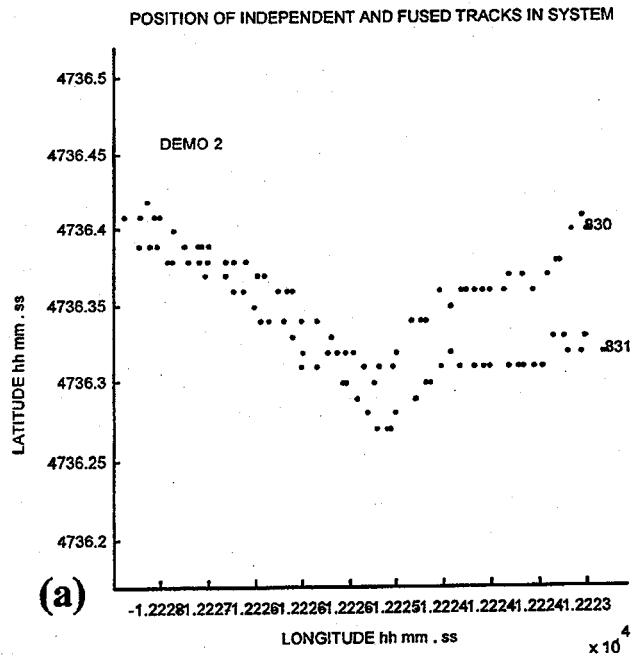


Figure 12. Overlapping Radar Coverage On A Single Vessel: (a) no fusion applied and (b) fusion applied

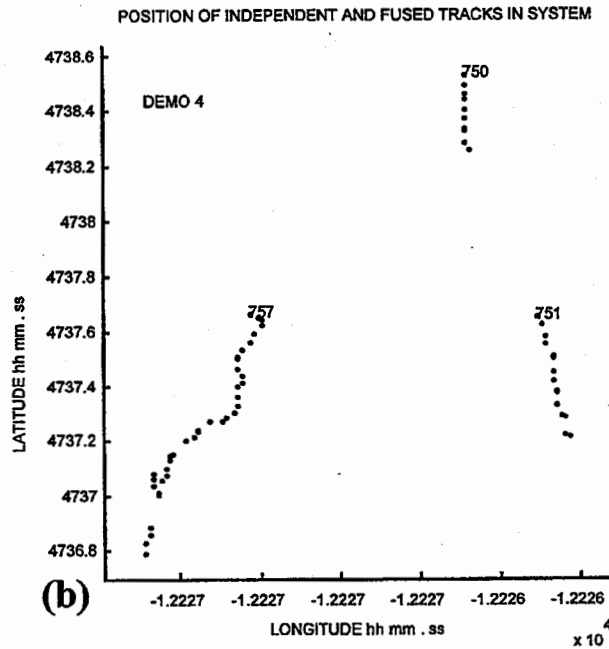
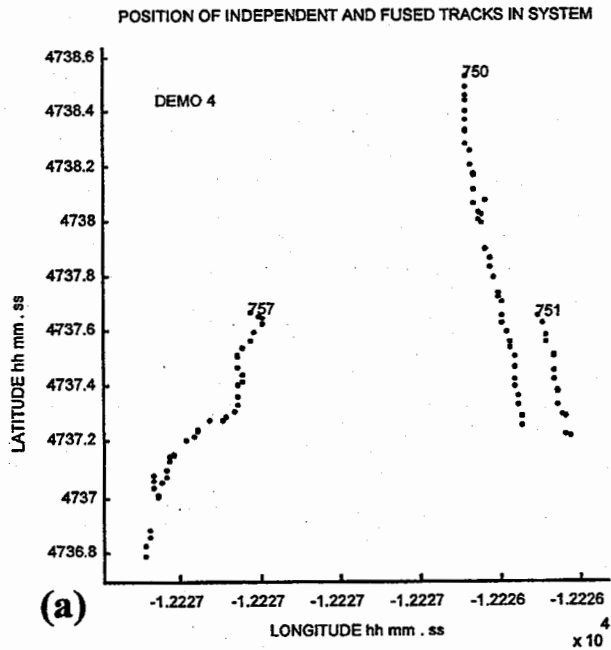


Figure 13 Overlapping Radar Coverage On A Single Vessel With An Independent Vessel:
 (a) no fusion applied and (b) fusion applied

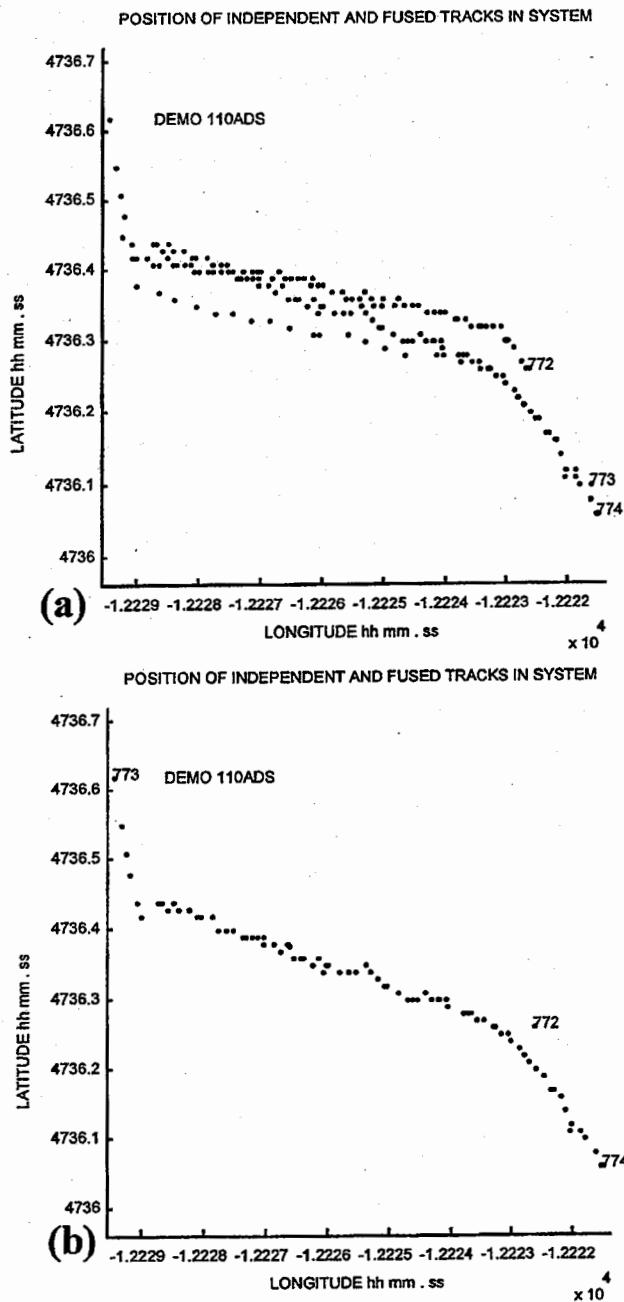


Figure 14. Overlapping Radar and ADS Coverage on a Single Vessel: (a) no fusion applied and (b) fusion applied

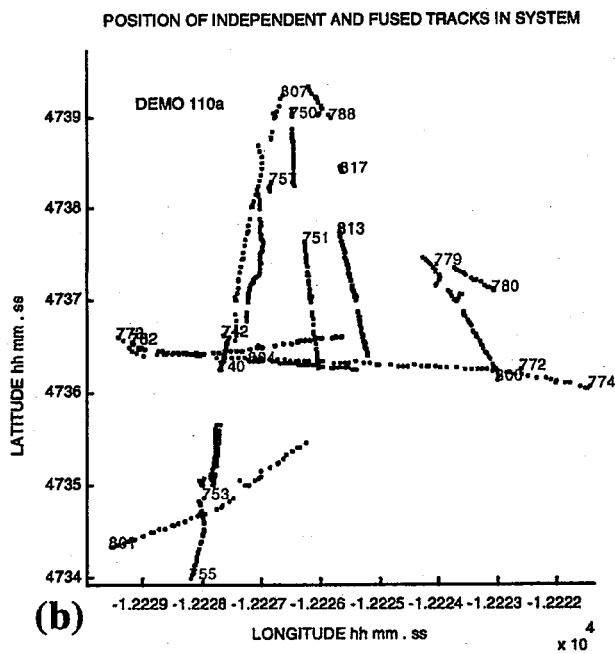
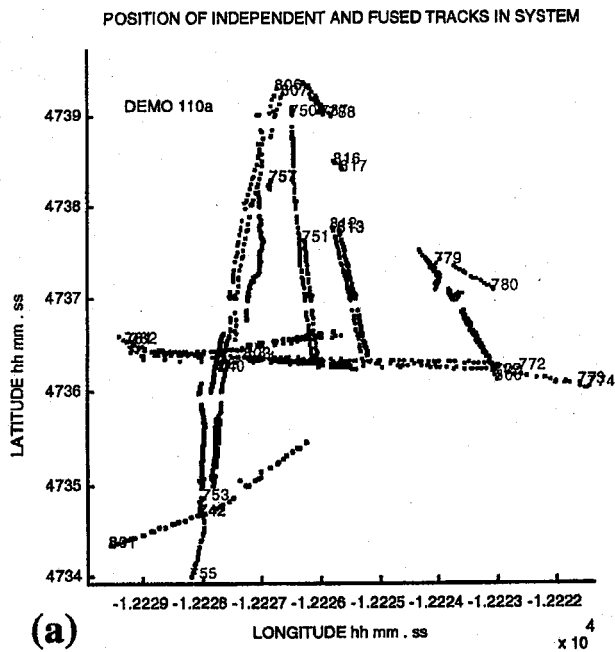


Figure 15. Overlapping Radar and ADS Coverage on Multiple Tracks: (a) no fusion applied and (b) fusion applied

VI. CONCLUSION

This thesis developed an algorithm to fuse redundant observations due to multiple sensor (type and location) coverage in order to provide a significant reduction in duplicate track information provided to the Vessel Traffic Services (VTS) operator displays. This **proof-of-concept** algorithm is a continuation of the work reported in [Ref. 1] and [Ref. 2]. The results presented are ready for final verification and validation by the USCG. The design of the algorithm allows acceptance of inputs from any type of sensor (radar, acoustic, GPS, system generated and manual tracks) as long as the basic decision criteria elements are provided. The result of this effort is a computationally efficient and cost effective software solution to a significant system deficiency that impacts greatly on overall waterway safety. The algorithm was tested with real data collected from the VTS system at Puget Sound in September 1996. The testing showed that the algorithm correctly fuses redundant sensor observations on the same vessel resulting in a significant reduction in the amount of unnecessary information presented to the VTS operator.

A. DISCUSSION OF RESULTS

The fusion algorithm performed as expected. The code extracted redundant track observations from multiple sensor sources on a single vessel and selected the most appropriate sensor to output the track information to the display. This resulted in a much improved representation of actual vessel movement in the harbors and waterways when compared to the current situation in the VTS System. The algorithm's current performance is limited by the number of attributes that could be used to determine association. Only *Latitude*, *Longitude*, *Course* and *Speed* were adopted to determine a level of "sameness" between vessels. The membership functions for *Latitude*, *Longitude* and *Speed* were triangular in nature as these attributes were considered to be reported accurately by both radar and ADS. The *Course* attribute is not reported with reasonable accuracy by radar when vessels are turning at reasonable speeds. The tracking algorithms in the radar

processors are the primary cause of this problem. Due to this problem, the membership function for *Course* is trapezoidal in shape allowing for a more generous association within a reasonable range but not outside a fixed value. ADS on the other hand reports *Course* very accurately. The algorithm can easily be modified to accept information from any sensor type as long as the specified attribute is available. The evaluation for a specified attribute can be turned off should it not be present in the data from a given sensor. Most importantly the algorithm can be modified to accept additional attributes which would further refine and improve the fusion decision making process.

B. SUGGESTIONS FOR FURTHER DEVELOPMENT

1. Additional Attributes

The performance of the algorithm can be enhanced by adding other attributes from which measures of similarity could be determined. *Size* and *TrackQuality* would appear to be likely candidates as this data could easily be extracted and/or reported from both radar and ADS sensors. These additional measures would allow for greater flexibility in applying the fusion process to a given set of track reports. With six measures to choose from, a weighting scheme relative to the importance of each membership function could be implemented. This is not feasible with the current four attributes as once you determine the relative "sameness" of position to each other via *Longitude* and *Latitude* you are left with just *Speed* and *Course*. *Speed* is a reasonably stable and accurate measure but the *Course* attribute has far too much variation for it to be dependable. Additional features would mitigate this problem somewhat. One other way to improve the algorithm without adding features is to improve the tracking capabilities of each radar's RSP to improve the accuracy and reliability of the *Course* data.

2. Data

Based on results to date, a program to collect more complete data sets, for further analysis and algorithm refinement, should be initiated. The data should include redundant observations from radar, ADS and SR sources over long periods of time. Collection should

be performed at all the different VTCs in order to provide thorough testing of the algorithm for as many scenarios as possible. This collection effort should be a long term commitment in order that all possible system configurations and atmospheric phenomena are encountered. A common approach and reporting method should be developed to enhance the quality of the data and thus its usefulness for testing and comparison purposes.

3. Membership Functions

The membership function design needs to be validated by statistical methods once large and varied data sets are available. This will optimize the design of the membership function for a given sensor and sensor suite within the applicable VTS System. Once these membership functions are validated for each type of sensor, the fusion algorithm could be made adaptive. For example, if two high resolution radars are reporting on a single target, they could use a much shorter base on the membership function providing a more accurate and timely decision making process. If three sensors of varying precision were tracking a given target, then the adaptive membership function would change to accommodate their characteristics. One step beyond this is to have the membership function shape adapt not only to the sensor type but also to the current statistics of the data. These changes would ensure that the dynamic conditions surrounding equipment performance and local atmospheric conditions could be factored into the fusion process in a seamless manner.

4. Validation and Verification

The proposed approach to the fusion of redundant observations due to multiple sensor (type and location) coverage on a single vessel needs to be validated through extensive statistical modeling and verified via on site testing at an operational VTC. The verification process could be accomplished with little difficulty as long as the data stream currently being received at the Tdbm could be simultaneously fed to a PC, running the program. This also could be accomplished by developing a JMCIS module for implementation into the VTS in order to carry out the fusion process in real time. Much of the overhead in the code is for data capture and display within MATLAB® and would not be necessary within the module which is encoded in the C language. Due to the

preponderance of IF-THEN-ELSE constructs within the algorithm, encoding in C would speed the execution time allowing for real time operation even if more attributes and other decision making criteria are introduced.

APPENDIX ALGORITHM CODE

This appendix lists the following code used in the fusion algorithm:

- **Fusex.m** - main fusion algorithm
- **getdatax.m** - data acquisition algorithm
- **timWin.m** - data set windowing algorithm
- **mstReTr4.m** - data set reduction algorithm

Table 1. Important Variable Names

VARIABLE	CONTENT	PARENT FUNCTION
ObsnMatrix	TRACK INFO FOR COMPLETE DATA SET	getdatax
MostRecentTrks	LAST TRACK UPDATE IN WINDOW	mstReTr4
tracksPresent	TRACK #S PRESENT IN Tdbm	mstReTr4
WindowedObsns	WINDOWED DATA SET	timWin
ColorOrder	TRACK COLOR SEQUENCING	fusex
startTime	TIME OF FIRST REPORT IN DATA SET	fusex
windowStart	TIME OF FIRST REPORT IN WINDOWED DATA SET	fusex
endTime	TIME OF LAST REPORT IN WINDOWED DATA SET	fusex
windowInc	SIZE OF SLIDING DATA WINDOW	fusex
FusionCandidates	TRACK PAIRS DEEMED FUSEABLE	fusex
CeaseReport	TRACKS THAT ARE DESIGNATED REDUNDANT	fusex
HitsToKeep	TRACKS ASSIGNED REPORTING RESPONSIBILITY	fusex
NumMostRecentTrks	NUMBER OF TRACKS PRESENT IN SYSTEM	fusex
LTDIF	DIFFERENCE IN LATITUDE BETWEEN A TRACK PAIR	fusex
LGDIF	DIFFERENCE IN LONGITUDE BETWEEN A TRACK PAIR	fusex
CRSDIF	DIFFERENCE IN COURSE BETWEEN A TRACK PAIR	fusex
SPDIF	DIFFERENCE IN SPEED BETWEEN A TRACK PAIR	fusex
TrackSensor	SENSOR TYPE REPORTING ON TRACK	fusex
TrackSensNum	LOCATION OF SENSOR REPORTING ON TRACK	fusex
TrksToPlot	FUSED AND INDEPENDENT TRACK DATA USED FOR OUTPUT	fusex
Lat_Window	LATITUDE DIFFERENCE FUSION THRESHOLD	fusex
Long_Window	LONGITUDE DIFFERENCE FUSION THRESHOLD	fusex
Course_Window	COURSE DIFFERENCE FUSION THRESHOLD	fusex
Speed_Window	SPEED DIFFERENCE FUSION THRESHOLD	fusex

```

% FUSION ALGORITHM
% fusex.m
%
% DESIGNED AND WRITTEN SEAN A MIDWOOD
%
% Original Concept by: Ian Glenn OCT 95
%
%
% Last Modified: 13 JULY 97
%
% Input:
% Track data from Tdbm
%
% Output:
% FUSION DECISION TO Tdbm
% INDEPENDENT AND FUSED TRACKS TO DISPLAY
%
% Design:
% load test data and run the Fusion algorithm(fusex.m) on it
%
% Calls:
% For Preprocessing
% getdatax.m
%
% For Fusion of Track Data
% timWin.m
% mstReTr.m
%
% For Plotting Results

```

```

global tracksPresent trackIndex
global TrckPres TrckHand TxtHand TrackInd ColorOrder

```

```

disp('
disp('          WELCOME TO THE USCG VTS FUSION ALGORITHM')
disp('          )
disp(' DEVELOPED AT THE NAVAL POSTGRAD SCHOOL, MONTEREY CALIFORNIA')
disp('          )
disp('          )
disp('          WORK PERFORMED BY SEAN MIDWOOD')
disp(' UNDER THE GUIDANCE OF PROF MURALI TUMMALA')
disp('          )
disp('          )
disp('          )
disp('          YOU WILL NOW BE ASKED TO ENTER THE VALUE')
disp('          )
disp(' OF TRACK PARAMETERS WHICH YOU WANT TO TEST ON')

```

```

disp('
disp(' THE TRACKS PRESENT IN THE VTS FOR FUSION PURPOSES.')
disp('
disp('
disp(' ENTERING 0.00 IS EQUIVALENT TO TURNING THE FUSION PROCESS OFF')
disp('
disp(' ***** THIS A NO FUSE SITUATION*****')
disp('
disp(' PRESS ANY KEY TO CONTINUE')
pause

```

```
%DECLARATION OF VARIABLES
```

```

Lat_Window=menu('CHOOSE DIFFERENCE IN MINUTES OF LATITUDE DEEMED
FUSEABLE','0.00','0.10','0.20','0.30','0.40','0.50','0.60');
Lat_Window=Lat_Window-1;
Lat_Window=Lat_Window/10;

```

```

Long_Window=menu('CHOOSE DIFFERENCE IN MINUTES OF LONGITUDE DEEMED
FUSEABLE','0.00','0.10','0.20','0.30','0.40','0.50','0.60');
Long_Window=Long_Window-1;
Long_Window=Long_Window/10;

```

```

Course_Window=menu('CHOOSE DIFFERENCE IN DEGREES OF HEADING DEEMED
FUSEABLE','0.00','5.00','10.00','15.00','20.00','25.00','30.00','35.00','45.00','50.00');
Course_Window=Course_Window-1;
Course_Window=Course_Window*5;

```

```

Speed_Window=menu('CHOOSE SPEED DIFFERENCE IN KNOTS DEEMED FUSEABLE','0.00','1.00',
'2.00','3.00','4.00','5.00','6.00','7.00','8.00','9.00');
Speed_Window=Speed_Window-1;

```

```
%Lat_Window=input(' ENTER FRACTION OF DEGREES OF LATITUDE CONSIDERED %FUSEABLE
= ')

```

```
%Long_Window=input(' ENTER FRACTION OF DEGREES OF LONGITUDE CONSIDERED
%FUSEABLE = ')

```

```
%Course_Window=input(' ENTER HEADING DIFFERENTIAL CONSIDERED FUSEABLE %=' ')

```

```
%Speed_Window=input(' ENTER SPEED DIFFERENTIAL CONSIDERED FUSEABLE = %')

```

```

TxtHand=[];
TrckPres=[];
TrackInd=[];
TrckHand = [];
tracksPresent=[];
trackIndex = [];

```

```
%ColorOrder = rand(100,3); %COLOURS USED FOR PLOT ROUTINE
```

```

ColorOrder=[ 0 0 0
             0 0 0
             1 1 1
             1 0 0
             0 1 1

```

```

001
101
010
111
100
011
001
101
010
111
100
011
001
101
010
111
100
011
001
101
010
111
100
011
001
101
010];

```

```
%GET DATA FROM Tdbm
```

```
%getdatax; %Read In Data Set
```

```
startTime = min(ObsnMatrix(:,7)); %start at the earliest time
%the observation matrix(ObsnMatrix)
```

```
%*****
```

```
windowSize =30000; %Number of seconds to include in
%window
```

```
%*****WARNING: THIS VALUE WILL HAVE TO BE ADJUSTED UP TO HANDLE*****
%*****LATENCY[LARGE TIME GAPS] IF YOU ARE USING LARGE FILES*****
%*****
```

```
windowStart=startTime + 3; %Take first 3 seconds of data
endTime = max(ObsnMatrix(:,7)); %Run to end of ObsnMatrix
```

```
% PLOT SET UP DETERMINED FROM MAX/MIN LATS/LONGS IN SYS
```

```
figure(1),clf
colordef black
axis([min(ObsnMatrix(:,2))-1,max(ObsnMatrix(:,2))+1),...
min(ObsnMatrix(:,1))-1,max(ObsnMatrix(:,1))+1)])
```

```

axis('on')
title('POSITION OF INDEPENDENT AND FUSED TRACKS IN SYSTEM')
xlabel('LONGITUDE hh mm . ss')
ylabel('LATITUDE hh mm . ss')
%axis square
%axis equal

%DATA WINDOW SIZE ADJUSTABLE DEPENDING ON HOW MUCH DATA YOU WANT TO
%PROCESS AND UPDATE RATES OF YOUR SENSORS
%*****
%*****CRITICAL PARAMETER*****
for window = windowStart:15:endTime %Slide the window in adjustable xx
    %second increments
%*****

%***** WINDOWING*****

%Once the data is available, it is windowed to extract the relevant %tracks in the specified time interval by
timWin.m

WindowedObsns = timWin(ObsnMatrix>window>windowSize);

% The next step is to extract the most
% recent observation on the tracks that are in the data 'window'

[MostRecentTrks] = mstReTr4(WindowedObsns);
%*****
Dummy=[0000 0000 0000 0000 0000 0000 0000 0000 0000 0000;0001 0001 0001 0001 0001 0001 0001 0001
0001 0001 0001];

MostRecentTrks = [Dummy;MostRecentTrks];

%KEEPS TWO TRACKS IN SYSTEM AT ALL TIMES TO PREVENT A NO DATA SITUATION
%*****
%*****
%*****FUSION ALGORITHM*****

%*****MEMBERSHIP FUNCTION ATTRIBUTES*****
%THESE CAN BE MADE ADAPATABLE IN THE FUTURE BY FIRST DETERMINING WHAT
%SENSORS ARE REPORTING ON A POTENTIAL REDUNDANT TARGET(S) THEN CHANGING
%THE ATTRIBUTES TO MATCH THE BASES OF THE MEMBERSHIP FUNCTIONS TO THE
%CAPABILITIES OF THE SENSOR. FOR EXAMPLE AN I/J-BAND NAV RADAR WILL HAVE
%MUCH BETTER RANGE AND BEARING RESOLUTION THAN A D-BAND RADAR. DGPS AND
%GPS LIKEWISE. THE BASE OF EACH MEMBERSHIP FUNCTION WOULD ALWAYS DEFAULT
%TO THE LEAST ACCURATE SENSOR.

%Lat_Window=.25;      % Fraction of a degree of latitude considered fuseable          % (500
yd seperation)
%Long_Window=.25;    % Fraction of a degree of longitude considered          %
fuseable (500 yd seperation)

```

```
%Course_Window=5.0;% Course difference considered fuseable
%Speed_Window= 5.0;% Speed difference considered fuseable
```

```
%MORE FUSION DECISION CRITERIA ARE EASILY ADDED AS THEY BECOME %AVAILABLE,
RELIABLE AND WELL DEFINED WITHIN YOUR SYSTEM. SIZE AND %TRACK QUALITY ARE
PRIME EXAMPLES OF THIS
```

```
%*****
```

```
FusionCandidates=[];           %Candidate matrix
CeaseReport=[];               %Update Tdbm as required
HitsToKeep=[];                %Suoerior Track #
NumMostRecentTrks=[];%       %Number of tracks in analysis window
```

```
%*****SEQUENTIAL TRACK ANALYSIS TO DETERMINE FUSION CANDIDATES****
```

```
ref_track =[];
compare_track=[];
```

```
for ref_track = 1:length(MostRecentTrks(:,1))
for compare_track = ref_track+1:length(MostRecentTrks(:,1))
```

```
NumMostRecentTrks=length(MostRecentTrks(:,1));
```

```
for n=1:length(NumMostRecentTrks)
if (NumMostRecentTrks >1)
```

```
    if (abs(MostRecentTrks(ref_track,1)-
        MostRecentTrks(compare_track,1))<=Lat_Window)
```

```
    LTDIF=(abs(MostRecentTrks(ref_track,1)-        MostRecentTrks(compare_track,1)))
```

```
        if (abs(abs(MostRecentTrks(ref_track,2))-
            abs(MostRecentTrks(compare_track,2)))<=Long_Window)
```

```
        LGDIF=abs(abs(MostRecentTrks(ref_track,2))-        abs(MostRecentTrks(compare_track,2)))
```

```
Diff=abs(MostRecentTrks(ref_track,3)-MostRecentTrks(compare_track,3));
```

```
    if Diff > 180           %THIS CODE WILL HANDLE HEADINGS
        Diff=360-Diff;     %THAT ARE CLOSE TO 000 AND PERFORMS THE
    else Diff=Diff;        %REQUIRED WRAP AROUND OF 000 TO 360
    end
```

```
    if Diff<=Course_Window
        %if (abs(MostRecentTrks(ref_track,3)-        %
            MostRecentTrks(compare_track,3))<=Course_Window)
```

```
    CRSDIF=Diff % (abs(MostRecentTrks(ref_track,3)-        %
        MostRecentTrks(compare_track,3)))
```

```
        if (abs(MostRecentTrks(ref_track,4)-
            MostRecentTrks(compare_track,4))<=Speed_Window)
```

```
        SPDIF=(abs(MostRecentTrks(ref_track,4)-        MostRecentTrks(compare_track,4)))
```

```
%pause
```

```
*****BUILD CANDIDATE MATRIX*****
```

```
FusionCandidates=[FusionCandidates;MostRecentTrks(ref_track,6),MostRecentTrks(ref_track,9),MostRecentTrks(ref_track,10),MostRecentTrks(compare_track,6),MostRecentTrks(compare_track,9),MostRecentTrks(compare_track,10)]
```

```
end % Lat  
end % Long  
end % Course  
end % Speed  
end %  
  
end  
end % inner for loop ref_track  
end % outer for loop compare_track
```

```
*****CHOOSE THE SUPERIOR SENSOR*****
```

```
%RADAR=1 IS SUPERIOR TO ADS=2 WHICH IS SUPERIOR TO SR=3*****
```

```
%THESE SHOULD BE OPERATOR AND OR SYSTEM INPUTS DEPENDING ON THE TYPES OF  
%SENSORS REPORTING, THEIR RELIABILITY, LATENCY AND PERSISTANCE.
```

```
if ((NumMostRecentTrks > 1) & (isempty(FusionCandidates)==0) )
```

```
for i =1:length(FusionCandidates(:,1))
```

```
Track(i)=FusionCandidates(i,1);  
TrackSensor(i)=FusionCandidates(i,2);  
TrackSensNum(i)=FusionCandidates(i,3);  
SimTrack(i)=FusionCandidates(i,4);  
SimTrackSensor(i)=FusionCandidates(i,5);  
SimTrackSensNum(i)=FusionCandidates(i,6);  
if (i > 1)  
if (Track(i) == Track(i-1))  
if (SimTrackSensor(i) < SimTrackSensor(i-1))  
FusionCandidates(i-1,:)= [0 0 0 0 0 0];  
elseif (SimTrackSensor(i) > SimTrackSensor(i-1))  
FusionCandidates(i,:)= [0 0 0 0 0 0];  
elseif (SimTrackSensNum(i) < SimTrackSensNum(i-1))  
FusionCandidates(i-1,:)= [0 0 0 0 0 0];  
elseif (SimTrackSensNum(i) > SimTrackSensNum(i-1))  
FusionCandidates(i,:)= [0 0 0 0 0 0];  
else  
FusionCandidates(i-1,:)= [0 0 0 0 0 0];  
end  
end  
end  
end
```

```
%TELL Tdbm TO CEASE REPORT ON THAT TRACK ID NUMBER
```

```
if (TrackSensor(i) < SimTrackSensor(i))
```

```
    CeaseReport=[CeaseReport;SimTrack(i)];
```

```
    HitsToKeep=[HitsToKeep;Track(i)];
```

```
% IF THE SENSOR TYPE IS THE SAME YOU MUST CHOOSE THE PREFERED SITE  
% SITE 1 IS SUPERIOR TO SITE 2 ETC BUT THIS SHOULD BE BASED ON SENSOR  
% CAPABILITIES AND CURRENT PERFORMANCE
```

```
elseif (TrackSensor(i) == SimTrackSensor(i))
```

```
    if (TrackSensNum(i) < SimTrackSensNum(i))
```

```
        CeaseReport=[CeaseReport;SimTrack(i)];
```

```
        HitsToKeep=[HitsToKeep;Track(i)];
```

```
    else
```

```
        CeaseReport=[CeaseReport;Track(i)];
```

```
        HitsToKeep=[HitsToKeep;SimTrack(i)];
```

```
    end
```

```
else
```

```
    CeaseReport=[CeaseReport;Track(i)]
```

```
    HitsToKeep=[HitsToKeep;SimTrack(i)];
```

```
end
```

```
end
```

```
end
```

```
HitsToKeep;    %UPDATE REPORT MATREX
```

```
VesselsInProximity=[CeaseReport;HitsToKeep];
```

```
VesselsInProximity=sort(VesselsInProximity);
```

```
TracksInSystem=MostRecentTrks(:,6);%TRACKS IN ANALYSIS WINDOW
```

```
% WE NOW HAVE TO INCLUDE TRACKS THAT ARE OFF THEIR OWN SO WE CAN SEND A  
% COMPLETE UPDATE REPORT TO THE Tdbm AND THUS THE DISPLAYS
```

```
UpdateReport=[HitsToKeep];
```

```
for i=1:length(MostRecentTrks(:,6))
```

```
    count=0;
```

```
        for j=1:length(VesselsInProximity)
```

```
            if (MostRecentTrks(i,6)~=VesselsInProximity(j,1))
```

```
                count=count+1;
```

```
            end
```

```
        end
```



```

        if (count==length(VesselsInProximity))
            UpdateReport=[UpdateReport;MostRecentTrks(i,6)];
        end

    end

UpdateReport=sort(UpdateReport);%COMPLETE UPDATE MATRIX OF INDEPENDENT
                                %AND FUSED TRACKS ID NUMBERS

% WE NOW HAVE TO BUILD A MATRIX THAT CONTAINS ALL THE PARAMETERS OF
% TRACKS THAT ARE TO REPORTED TO THE Tdbm AND DIPLAY

%FIND THE INDEX OF EACH OF THE TRACKS IN THE UPDATE

x=[];

Z=length(UpdateReport);

for M=1:Z;

x(M)=find(MostRecentTrks(:,6)==UpdateReport(M));

end

%BUILD UPDATE MATRIX BY EXTRACTING TRACK PARAMETERS FROM ANALYSIS WINDOW

InfoToPlot=MostRecentTrks([x],:);

% SORT BASED ON TRACK ID #

[TracksSorted,trackIdx]=sort(InfoToPlot(:,6));

% FIND IF THERE ARE ANY REDUNDANT TRACKS

difference = diff([TracksSorted;max(TracksSorted)+1]);
trackCount = diff(find([1;difference]));
TrckPres= TracksSorted(find(difference));

% SELECT THE MOST CURRENT REPORT IF A REDUNDANT TRACK EXISTS

TrksToPlot=[];
    for i=1:length(TrckPres)
        TrksToPlot=[TrksToPlot;...
                    InfoToPlot(trackIdx(sum(trackCount(1:i))),:);
    end
    DisplayUpdate=TrksToPlot(:,6)

%CHECK FOR REDUNDANCY
%IF A TRACK # IS PRESENT IN BOTH CeaseReport AND HitsToKeep
%ELIMINATE IT FROM H

```

```

for j=1:length(TrksToPlot(:,6))
    for k=1:length(CeaseReport)
        if (TrksToPlot(j,6)~=CeaseReport(k,1))
            TrksToPlot(j,:)= [0,0,0,0,0,0,0,0,0];
        end
    end
end
end
end
DisplayUpdate

```

```

%*****PLOT ROUTINE*****
%*****

```

```

% First time through, create the tracks

```

```

if isempty(TrckHand)

```

```

    for n = 1:size(TrksToPlot,1)
        TrckHand(n) = line(TrksToPlot(n,2),TrksToPlot(n,1),...
            'linestyle','o','Color',ColorOrder(n,:),...
            'Erasemode','none',...
            'MarkerSize',2);
        TrackInd(n,:) = [TrckPres(n), n];
        TxtHand(n) = text(TrksToPlot(n,2),TrksToPlot(n,1),...
            int2str(TrckPres(n)),...
            'Color', ColorOrder(n,:),...
            'FontSize',10);
    end

```

```

else

```

```

    for m = 1 : size(TrksToPlot,1)
        n = find(TrksToPlot(m,6)~=TrackInd(:,1)); % Index of current
        if isempty(n);

```

```

            n = size(TrackInd,1)+1;
            TrckHand(n) = line(TrksToPlot(m,2), TrksToPlot(m,1),...
                'linestyle','o','Color',ColorOrder(n,:),...
                'Erasemode','none',...
                'MarkerSize',2);
            TrackInd(n,:) = [TrksToPlot(m,6), n];
            TxtHand(n) = text(TrksToPlot(m,2),TrksToPlot(m,1),...
                int2str(TrackInd(n,1)),...
                'Color',ColorOrder(n,:),...
                'FontSize',10);
        end

```

```

    else

```

```

        set(TrckHand(n),'Xdata',[get(TrckHand(n),'Xdata'),TrksToPlot(m,2)],...
            'Ydata',[get(TrckHand(n),'Ydata'),TrksToPlot(m,1)]);
        set(TxtHand(n),'Position',[TrksToPlot(m,2),TrksToPlot(m,1)]);
    end

```

```
end  
end  
end
```

```
drawnow      % OUTPUT TO DISPLAY
```

```
end          % END OF FUSION ALGORITHM GO BACK AND GET THE NEXT SET OF  
            % DATA FROM Tdbm
```

%DATA CAPTURE ALGORITHM

%getdatax.m

%THIS FUNCTION TAKES IN DATA SUPPLIED BY THE USCG AND PUTS IT IN A %FORMAT THAT CAN BE USED BY THE FUSION ALGORITHM. THE OUTPUT IS AN %OBSERVATION MATRIX WHICH SIMULATES THE TDBM.

```
BVesselName = '          '; % 26 spaces for padding

%BTrackStatus = '    '; % 5 spaces for padding

% Initialize Storage vectors

VesselName = [];
UTC = [];
TrackStatus = [];
TrackIDNumber = [];
SensorTrackNumber = [];
TrueCourse = [];
Speed = [];
Latitude = [];
Longitude = [];
Size = [];
TrackQuality = [];

filename = input('Enter file name » ','s');

% -----Start reading the file -----

fid = fopen(filename,'r'); % Read only

st = fgets(fid); % Get first line

while st ~= [-1]; % Check for end-of-file N = 1:10

Cloc = findstr(st,','); % Finds delimiter

VesselName = [VesselName; st(1:Cloc(1)-1),BVesselName(1:26-length(st(1:Cloc(1)-1)))];

UTC = [UTC; str2num(st(Cloc(1)+1:Cloc(2)-1))];

TrackStatus = [TrackStatus; str2num(st(Cloc(2)+1:Cloc(3)-1))];

%BTrackStatus(1:5-length(st(Cloc(2)+1:Cloc(3)-1)));

TrackIDNumber = [TrackIDNumber; str2num(st(Cloc(3)+1:Cloc(4)-1))];

SensorTrackNumber = [SensorTrackNumber;str2num(st(Cloc(4)+1:Cloc(5)-1))];

TrueCourse = [TrueCourse; str2num(st(Cloc(5)+1:Cloc(6)-1))];
```

```
Speed = [Speed; str2num(st(Cloc(6)+1:Cloc(7)-1))];
```

```
Latitude = [Latitude; str2num(st(Cloc(7)+1:Cloc(8)-1))];
```

```
Longitude = [Longitude; str2num(st(Cloc(8)+1:Cloc(9)-1))];
```

```
Size = [Size; str2num(st(Cloc(9)+1:Cloc(10)-1))];
```

```
TrackQuality = [TrackQuality; str2num(st(Cloc(10)+1:Cloc(10)+2))];
```

```
st = fgets(fid);
```

```
end
```

```
% BUILD DATA BASE OF ALL OBSERVATIONS PRESENT IN TDBM
```

```
ObsnMatrix=[Latitude Longitude TrueCourse Speed Size TrackIDNumber UTC TrackQuality TrackStatus  
SensorTrackNumber];
```

```
function WindowedObsns = timWin(OrigObsnMatrix,timeNow>windowLength)
%function WindowedObsn = timWin(OrigObsnMartix,timeNow>windowLength)
%
%   by:           Ian Glenn
%   modified by:  Sean Midwood
%
%   Created:      10 Sep 95
%   Last Modified: 14 Jul 97
%
%
%   Input:
%       OrigObsnMatrix: original obsn matrix to be windowed
%       timeNow:         time (in sec) to window from
%       windowLength:   number of seconds to window back in time
%
%   Output:
%       Data window to operate on
%
%   Design:
%       find the tracks that meet the time criteria
%       and create the new windowed matrix to be used
```

```
rows=find(OrigObsnMatrix(:,7)<=timeNow & OrigObsnMatrix(:,7)>=(timeNow-windowLength));
```

```
WindowedObsns=OrigObsnMatrix(rows,:);
```

```

function [MostRecentTrks] = mstReTr4(ObsnMatrix)
%
% function [MostRecentTrks] = mstReTr4(ObsnMatrix)
%
% by:          Ian Glenn
% modified by: Sean Midwood
%
% Created:     29 AUG 95
% Modified:    14 JUL 97
%
% Input:
%           I1 - ObsnMartix:original obsn matrix to be windowed
%
%
% Output:
%           O1 - MostRecentTrks
%
%
% Design:
%
% This function determines which tracks are present and how many
% in a given set of tracks and returns the matrix with the most % recent observations

%disp('Call MstReTr')
global tracksPresent trackIndex

MostRecentTrks=[];

[TracksSorted,trackIndx]=sort(ObsnMatrix(:,6));% Sort by TrackIDNumber

% Find the redundancy in a vector x

difference = diff([TracksSorted;max(TracksSorted)+1]);
trackCount = diff(find([1;difference]));
tracksPresent= TracksSorted(find(difference));

% It is now easy to select the latest value for each track with
% trackIndx(trackCount(1))

for i=1:length(tracksPresent)
    MostRecentTrks=[MostRecentTrks;...
                    ObsnMatrix(trackIndx(sum(trackCount(1:i))),:);
end

```


LIST OF REFERENCES

1. Glenn, Ian N., "Multilevel Data Association for the Vessel Traffic Services System and Joint Maritime Command Information System," Master's Thesis, Naval Postgraduate School, Monterey, California, December 1995.
2. Ruthenberg, Thomas M., "Data Fusion Algorithm for the Vessel Traffic Services System: A Fuzzy Associative System Approach," Master's Thesis, Naval Postgraduate School, Monterey, California, March 1995.
3. Inter-National Research Institute (INRI), *Tdbm Service Application Programmer's Interface (API) For the Unified Build (UB) Software Development Environment (SDE)*, SPAWARSYSCOM SDE-API-TDBM-2.0.11.5, March 31, 1994.
4. Inter-National Research Institute (INRI), *Functional Description Document for Automated Dependent Surveillance (ADS)*, July 25, 1995.
5. Technical Information Exchange Meeting between Inter-National Research Institute, USCG and NPS, Reston, Virginia, November 3, 1995.
6. Telephonics, *MTE-2000 Marine Target Extractor Technical Manual*, Part No. 523854, Farmindale, New York, April 1, 1995.
7. USCG Home Page, <http://www.dot.gov/dotinfo/uscg/factfile/gps.html>
8. "Software Design Document for JMCIS, ADS Proof of Concept (Phase I)," Inter-National Research Institute (INRI), Reston, Virginia, December 1995.
9. "Functional Description Document for the JMCIS ADS VER 1.1," Inter-National Research Institute, Reston, Virginia, December 1995.
10. Hall, D.L., Llinas J., "An Introduction to Multisensor Data Fusion," *Proceedings of the IEEE*, Vol. 85, No. 1, January 1997.
11. Waltz, E. and Llinas, J., *Multi-Sensor Data Fusion*, Artech House Inc., Boston, Massachusetts, 1990.
12. Hall, David L., *Mathematical Techniques in Multisensor Data Fusion*, Artech House Inc., Norwood, Massachusetts, 1992.
13. Kandel, A., *Fuzzy Mathematical Techniques with Application*, Addison - Wesley, Reading, Massachusetts, 1986.

14. Personal Communication, Naval Postgraduate School, and USCG, Meeting, Portsmouth, Virginia, 9 July 1997.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2. Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5121	2
3. Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4. Professor Murali Tummala, Code EC/Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	3
5. Professor Roberto Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
6. Professor Herschel H. Loomis, Code EC/Lm Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
7. Major Ian N. Glenn DASPM 4 National Defence Headquarters Ottawa, Ontario Canada K1A 0K2	1
8. CDR Michael Linzey Command Control Engineering Center 4000 Coast Guard Blvd. Portsmouth, VA 23703-2199	1

- | | | |
|-----|---|---|
| 9. | LCDR Gordon Weekes
Command Control Engineering Center
4000 Coast Guard Blvd.
Portsmouth, VA 23703-2199 | 1 |
| 10. | LCDR Sean A. Midwood
395 Daly Ave.
Ottawa, Ontario
Canada K1N 6H1 | 2 |