Integrated End-to-End Radar Signal & Data Processing With Over-arching Knowledge-Based Control

Gerard T. Capraro
Capraro Technologies, Inc.
311 Turner Street – Suite 410
Utica, NY 13501
USA
gcapraro@caprarotechnologies.com

Abstract

This paper provides information related to integrating Knowledge Based (KB) techniques within the filtering, detection, tracking and target identification portions of an airborne radar's processing chain. We will present multiple information sources and how they can be used to enhance a radar's performance for end-to-end signal and data processing.

Introduction

In our previous paper we presented material for understanding some of the basic elements regarding knowledge bases and artificial intelligence (AI). In this paper we wish to present a design of an intelligent airborne radar system that processes information from the end-to-end, i.e. filter, detector and tracking stages of a surveillance radar. Can we build new radar systems that can dynamically change its processing given information from other sensors, outside sources, weather data, etc.? We believe that we can. The computing clock rates for computers have been doubling approximately every 18 months. Today's commercial off the shelf computers have clock rates exceeding 3 GHz. We believe that the computing power is available to insert sophisticated "rules/logic" within radar signal and data processing.

The following section will pick up where we left off in our first paper, dealing with ontologies. A global view of interfacing multiple platforms of sensors and the integration of sensors on one platform will be discussed. The next section will describe the major knowledge base components of an airborne intelligent radar system (AIRS). The next section will provide an overview of how the AIRS processes data within different states. The following section discusses the issues with adapting an AIRS architecture for sensors on board unmanned air vehicles (UAV). The following section will provide a knowledge base tracking algorithm with memory thereby providing information helpful for target identification and terrain resolution. The last section provides our summary.

A Global View

The performance of our sensor systems can be enhanced by dynamically controlling a sensor's algorithms dependent upon a changing environment. The sharing of information in real time with other sensors is also a major plus. It has been shown in this lecture series that if an airborne radar system knows about certain features of the Earth (e.g. land sea interfaces) and its surroundings then it can use this information intelligently and increase its performance. A radar system can perform better with information from other sensors, e.g. sensor fusion. It could perform better if it knew where potential jammers were located and their characteristics.

Capraro, G.T. (2006) Integrated End-to-End Radar Signal & Data Processing With Over-arching Knowledge-Based Control. In *Knowledge-Based Radar Signal and Data Processing* (pp. 8-1 – 8-24). Educational Notes RTO-EN-SET-063bis, Paper 8. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

However, if an airborne radar is going to share and receive information from multiple sources then it must be able to communicate and understand the information. A solution for the exchange of information between heterogeneous sensors is for each sensor to publish information based upon shared ontologies. In this manner when a sensor publishes its track data multiple sensors receiving this information will be able to interpret its contents without ambiguity. Accomplishing this will require that certain basics be established. We must have an accepted method of defining the Earth's geometry such that every element on the Earth, air or space's positions are all defined within the same coordinate system. That each element is time synchronized with the same clock and all communications are time stamped.

Each transmission of information between sensors must depict its time and its coordinates. In addition if it is sharing track or target data it must specify their unique identifier, its velocity, pitch, yaw, and role and meta data describing the transmitted raw data along with encryption/decryption keys. The unique identifier will allow the receiving sensor to acquire, within its resident database management system (DBMS), all of the sender's radar characteristics. The description of these data can be defined by ontologies such that all the sensor platforms will correctly understand the information provided. Sensor characteristics include such things as nomenclature, power output, bandwidth, frequency, antenna pattern, pulse width, pulse repetition frequency (PRF), etc. Platform characteristics as to the position of the antenna on the platform, number of elements, the pattern of the elements, the pointing vector of the radar, etc. We need an ontology for defining these data and numerous rules so that the information published by any sensor can be understood correctly by the receiving sensor to perform functions such as sensor fusion, track correlation, and target identification.

Sharing information between sensors on the same platform is also required, especially if one or more sensors are adaptively changing its waveform parameters to meet the demands of a changing environment. Figure 1 depicts a hypothesized intelligent sensor system. Each of the sensors has its own signal and data processing functional capability. In addition to this capability we have added an intelligent processor to address fusion between sensors, communication between sensors, and control of the sensors. The goal is to be able to build this processor so that it can interface with any sensor and communicate with the other sensors using ontological descriptions via the intelligent platform network. The intelligent network will be able to coordinate the communications between the sensors on board and to off platform sensor systems. There are approaches we can exploit to build this system by using fiber optic or wire links on board the platform. Radio frequency (RF) links using Bluetooth or 802.11 technologies can be exploited for linking these sensors on board the platform. Between platforms other technologies may be exploited such as mobile internet protocol over RF communications links. The communications issues need to be addressed for the sharing of information and for minimizing the potential of electromagnetic (EM) fratricide. The intelligent platform should determine if there is EM interference (EMI) potential when a sensor varies their antenna's main beam pointing vector, or changes its PRF and may thereby cause interference to a receiving sensor. Rather than have each sensor on a platform operate as an independent system we need to design our platform as a system of sensors with multiple goals managed by an intelligent platform network that can manage the dynamics of each sensor to meet the common goal(s) of the platform. This is one of the major goals we are pursuing under our sensors as robots initiative. This initiative is addressing attended and un-attended sensor platforms.

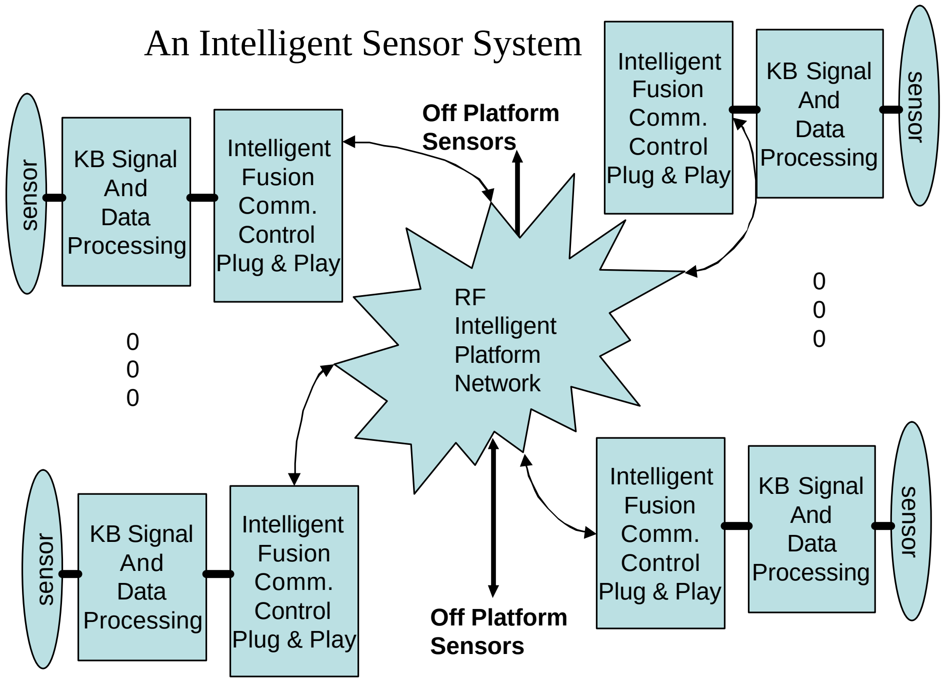
An Intelligent Sensor System
sensor
KB Signal And Data Processing
Intelligent Fusion Comm. Control Plug & Play
Off Platform Sensors
RF Intelligent Platform Network
0 0 0
Off Platform Sensors

Figure 1. An Intelligent Sensor System

An Airborne Intelligent Radar System (AIRS)

The KB signal and data processing portion shown in figure 1 may represent one radar sensor system. If this sensor system is built using knowledge based techniques then there exists intelligence to control its own processing. A modified design obtained from the KB Space Time Adaptive Processor (KBSTAP) effort (2) is shown in figure 2. In this section we will describe the major components of this knowledge base radar design. In the figure we have labeled the major components as processors with the knowledge base controller as the major integrator for communications and control of the individual processors. These processors operate independently and cooperatively. They can be implemented on a separate computer or on the same computer and operate as separate software processes. The knowledge base controller (KBC) receives information from many sources. Data about the radar, its frequency of operation, antenna configuration, where it is located on the aircraft, etc. is provided by the block labeled in figure 2, configuration information. The map data is preloaded before each mission for estimating clutter returns and for registering its location relative to the Earth and with other sensor platforms. It is also preloaded with its flight profile data and is updated continuously from the platforms navigation system. It also will receive information from the intelligence community both before a mission and throughout the mission. During flight, the KBC will receive information about weather, jammer locations, requests for information, discrete locations, fusion information, etc. We are assuming that the radar system is aboard a surveillance aircraft flying a known and repeatable path over the same terrain. Therefore it can learn by monitoring the performance of different algorithms over repeatable passes of terrain.

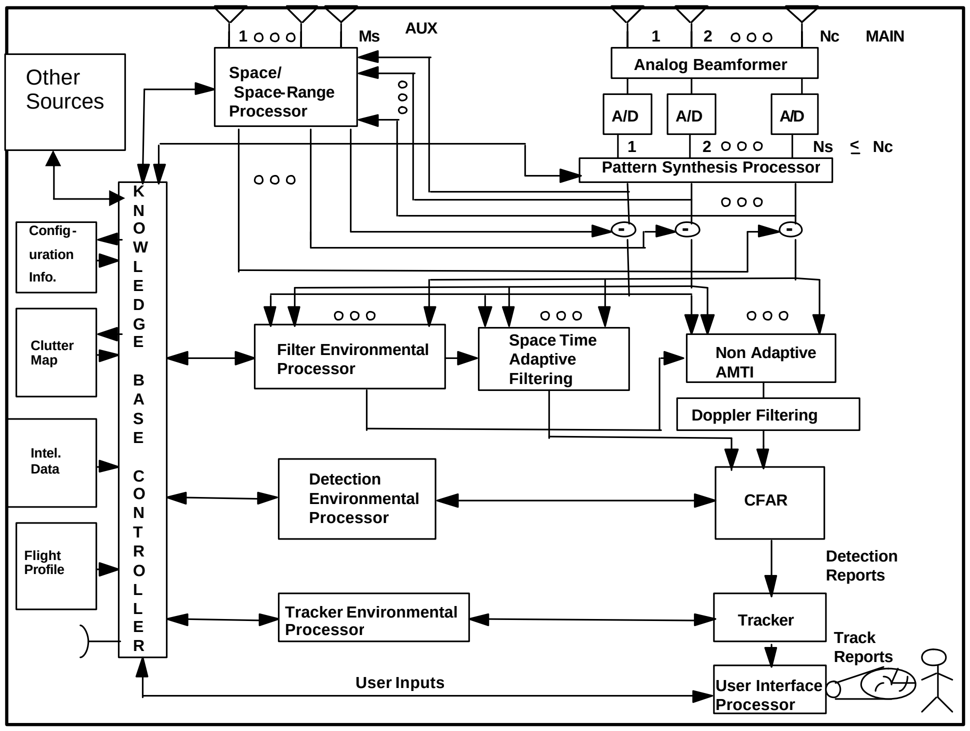

Figure 2. An AIRS Architecture

The KBC performs the overall control functions of the AIRS. It assigns tasks to all processors, communicates with outside system resources, and "optimizes" the system's global performance. Each individual processor "optimizes" its individual performance measures, e.g. signal-to-noise ratio and probability of detection. The tracker with the KBC, for example, "optimizes" the number of correct target tracks and "minimizes" the number of missed targets, incorrect tracks, and lost tracks. The KBC handles all interrupts from the User Interface Processor, assigns tasks to the individual processors based upon user requested jobs, generates information gathered from sources to enhance the performance measures of the individual processors, works with other sensors and outside sources for target identification, and provides the User Interface Processor periodic and aperiodic data for answering queries and requests from the user.

Space/Space Range Processor (SSRP), Pattern Synthesis Processor (PSP), Filter Environmental Processor (FEP) and KBC Interfaces

The KBC will provide geographical information e.g. it will periodically provide the direction the receiver is looking, clutter maps, the location of the emitter, locations of hot clutter jammers, locations of direct jammers or electromagnetic interference sources, and discretes. The KBC will also provide tasks to the SSRP, PSP and FEP. It will for example, task each of the sources of "interference" be reduced by a defined amount. Sources of interference will be prioritized. The SSRP, PSP, and FEP once tasked, will implement and control their own algorithms and processing. The processors will optimize the KBC's request given the number of available degrees of freedom and their physical operational constraints.

The KBC will provide control and operational requests based upon global optimization considerations and/or input directions from the user. For example, the user may want to execute multiple algorithms and compare their results. This may require parallel processing on the same set of data. The user may wish to restrict portions of algorithms from being executed e.g. the user may task the AIRS to compare the performance with and without pattern synthesis. These different tasks will require the KBC to direct the control of each processing stage to operate in a parallel processing mode. Figure 3 illustrates eight different parallel processing modes that will occur when restricting no more than two different algorithms per processing stage. This approach of executing parallel algorithms, as directed by the user, will allow AIRS to learn which algorithms perform better under identical conditions.

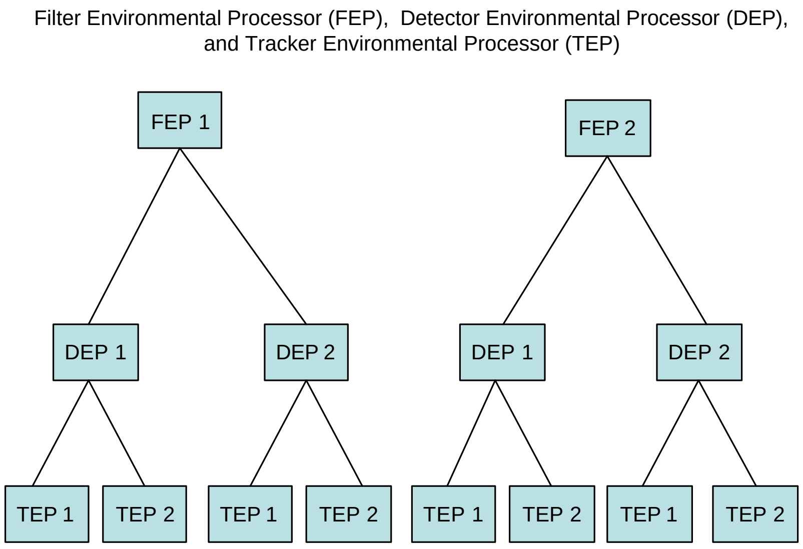
Filter Environmental Processor (FEP), Detector Environmental Processor (DEP), and Tracker Environmental Processor (TEP)
FEP 1
FEP 2
DEP 1
DEP 2
DEP 1
DEP 2
TEP 1
TEP 2
TEP 1
TEP 2
TEP 1
TEP 2
TEP 1
TEP 2

Figure 3. Parallel Algorithms

The results of the KBC's tasks will be reported to the KBC, as a joint or cooperative accomplishment of the three processors. The amount of interference cancellation obtained for each interference source will be reported by the FEP. The information will include the amount of dB attenuation per interference source, whitening, and gain loss. All three processors (SSRP, PSP, FEP) will report to the KBC, the algorithms used and their parameter values.

The three processors' general operating procedure is to use all of their available resources while attempting to exceed KBC tasks. If the resultant global performance measures are not met then the KBC can change the tasks to these processors during the next iteration.

Detection Environmental Processor (DEP) and KBC Interface

The KBC provides the DEP filter output data, clutter map data and results from the tracker such as the degree of belief or weights/importance of previously detected targets. This information allows the DEP to choose its models for the next iteration of data. For instance, the algorithm may adjust its threshold if a high priority target is entering a different clutter background.

The KBC directs the DEP through tasks as discussed in the previous stage. For instance, if the detection process was performed within the filtering stage then the KBC will either "shut down" the DEP or request it to run parallel processes on data obtained from the filter processor. Consider figure 4 where FEP1 incorporates detection and FEP2 does not.

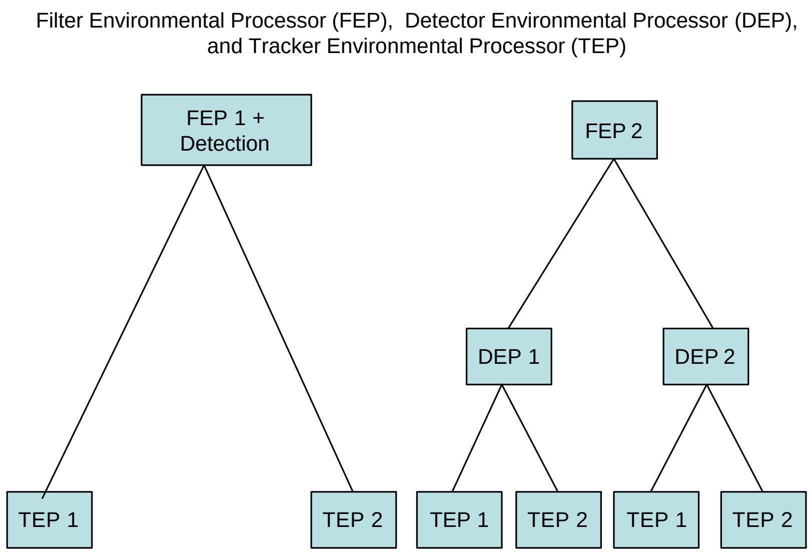
Filter Environmental Processor (FEP), Detector Environmental Processor (DEP), and Tracker Environmental Processor (TEP)
FEP 1 + Detection
FEP 2
DEP 1
DEP 2
TEP 1
TEP 2
TEP 1
TEP 2
TEP 1
TEP 2

Figure 4 Bypassing the Detection Processor

The results of the KBC's tasks will be communicated back to the KBC. Probability of detection, probability of false alarm, algorithms used, and their parameter values will be reported to the KBC.

<u>Tracker Environmental Processor (TEP) and KBC Interface</u>

The KBC provides data to the TEP that are not contained in the detection data provided by DEP e.g. priority of targets/tracks. The KBC provides control information to the TEP similarly as discussed above based upon parallel processes, choices of algorithms and their parameters, and any definitive requests made by the user.

The TEP will report back to the KBC for each process, each track's probability, the probability of missed tracks or lost tracks, and additional performance measures associated with the algorithms used and their parameter values.

<u>User Interface Processor (UIP) and the KBC Interface</u>

The KBC provides data and receives control from the user via the UIP. Directed by the user the KBC will task the Process Manager and Data Manager (not shown in figure 2) to pre-configure the computers and algorithms for each of the above processors for the next flight iteration or CPI. It will provide information related to intermediate results, performance measures, how AIRS arrived at its solutions, and assist the UIP in configuring the antenna and processors.

<u>Configuration Information and KBC Interface</u>

The exchange between the Configuration Information and the KBC contains for example data regarding the radar, the radar's location, antenna, and transmitter characteristics. Some of these data can be modified by the user and are pre-stored in the Data Manager and accessed via the UIP.

Clutter Map and KBC Interface

The Clutter Map is defined given the flight profile of the aircraft. This file contains those parameters required by the AIRS' algorithms obtainable from actual terrain files such as land use land clutter (LULC), digital elevation model (DEM) and digital line graph (DLG) databases. These data are provided by the US Geological Survey (USGS). These data will be stored in an environmental data file and accessed via the Data Manager along with clutter map data computed on the fly during flight.

Intelligence Data and KBC Interface

Intelligence community data are provided to the AIRS. These data may contain the location of jammers, a jammer's parameters, target parameters and a target's kinematics. These data will be used by different AIRS' algorithms and knowledge sources.

Flight Profile and KBC Interface

Flight profile data are stored and maintained in a database via the Data Manager. These data contain parameters required by AIRS' algorithms.

Antenna and KBC Interface

This antenna represents the communications link to outside sources for gathering and providing information during flight.

AIRS State Processing

AIRS is a dynamic system, i.e. it changes its processing dependent upon its goals and the environment. This section provides an overview of a hypothetical AIRS and its operation during changing conditions. AIRS' processing begins by loading its computers with pre-flight mission, intelligence, and terrain data. The process will go through four states; pre-flight, initial transient state, correlation/performance/assessment/learning state, and steady state. Steady state probably won't occur until the aircraft (A/C) flys at least one to two race tracks over the same area. The initial transient state will take 4 to 20+ CPIs before tracks can be formed and AIRS starts identifying interrogation-friend-or foe (IFF) related tracks. The intermediate state: to correlate discretes, objects, shadow regions, and jammers, evaluate performance measures and set thresholds, and deciding which objects require nulling, how much nulling, and when nulling should occur. Table 1 provides a brief description of the four different states and the functions of the KBC, its performance processor, and the three main radar intelligent system processors (filter, detector and tracker). We have partitioned the KBC into two processor functions: one to control the AIRS and one to monitor and report its performance throughout its different stages.

System States Versus Processors	**Pre-Flight**	**Initiate System & Initial Transient States (4 to 20+ CPIs)**	**Correlation, Assessment, Learning (1 to 2 Complete Tracks of a Defined Scene/Area)**	**Steady State**
K B Performance Processor	1- Locate and load all Potential Discretes, Clutter Boundaries, Shadow Regions, Jammers, Obstacles - Set System Parameters	6- Monitor System	11- Correlate Discretes, Clutter Boundaries, Shadow Regions, Potential Jammers, Obstacles - Evolve Rules - Insert Synthetic Targets - Measure Performances	16- Insert Synthetic Targets - Measure Performances - Change Rule Sets Accordingly
K B Controller	2- Locate and load all Potential Discretes, Clutter Boundaries, Shadow Regions, Jammers, Obstacles - Set System Parameters	7- Initiate System and Monitor	12- Correlate Discretes, Clutter Boundaries, Shadow Regions, Potential Jammers, Obstacles - Evolve Rules	17- Measure Performances - Change Rule Set Accordingly
Intelligent Filter Environmental Processor	3- Define initial settings and performance measure thresholds	8- Execute Non-STAP Algorithm - Compute No of Secondary Rings - Run NHD - Compute Beam Performance, Determine Null Weights - Determine STAP feasibility	13- Compute Number of Sec. Rings, Run NHD, Compute Beam Performance Measures, Set Nulls, Determine When and Where STAP is Feasible - Evolve Rules	18- Measure Performances - Change Rule Set Accordingly
Intelligent Detector Environmental Processor	4- Define initial settings and Thresholds for Pfa	9- Compute and Adjust Thresholds for Pfa	14- Compute Detections – Re-compute and Adjust Pfa Thresholds - Evolve Rules	19- Measure Performances - Change Rule Set Accordingly

Intelligent Tracker Environmental Processor	5- Locate all Potential Discretes, Clutter Boundaries, Shadow Regions, Jammers, Obstacles - Define initial settings and performance measure thresholds	10- Initiate Tracks - Compute Performance Measures (Number of Correct Tracks, Number of Dropped Tracks, Number of Incorrect Tracks)	15- Correlate FAA Data with Tracks - Compute Performance Measures - Number of Tracks, Number of Dropped Tracks, Number of Incorrect Tracks - Evolve Rules	20- Measure Performances - Change Rule Set Accordingly

Table 1. AIRS States Versus Processors

<u>1-2 Pre-Flight for KB Processors.</u>

The hypothesized location of discretes, clutter boundaries, shadow regions, potential jammers, and obstacles are loaded into AIRS. This can be performed in at least two ways.

1. Load the location of all these entities into one table and as AIRS begins learning it will find detections that it will try to correlate with entities in the table. As they are verified, their status will be changed from hypothesized to identified and their parameters updated accordingly. As new entities are found they will be entered into the table as hypothesized and when verified, with detections from more than one race track, they will be upgraded to identified.

2. Load all hypothesized entities into separate tables based upon their type (i.e. discretes, obstacles, shadow regions, aircraft, etc.) and as they are verified they will be marked identified and their parameters updated. As new entities are found they are placed into a general table and as they are verified they can be moved to their proper table.

The classification and storage of the different entities can be done in many ways. Consider the following relations as one example.

Road Traffic (Road ID #, LL1, LL2, LL3, . . . , LLn, Priority, Confidence (= 0 when first loaded, i.e. hypothesized))

LLi implies latitude and longitude of points on the Earth that defines a straight line approximation of a road.

Discretes (Discrete ID #, LL1, LL2, …, Priority, Confidence (= 0 when first loaded))

A discrete may require one or more latitude and longitude points to describe its position, e.g. a steel bridge.

Clutter Types (Clutter Type ID #, LL1, LL2, LL3, . . . , LLn, Priority, Confidence (= 0 when first loaded))

The location points define the boundaries for different clutter types, such as urban, forest, ocean, etc. Every point within a boundary is modeled as the same type of clutter. This describes the homogeneous clutter model for choosing secondary data for STAP.

For shadow or obstacle type (Shadow/Obstacle ID #, LL1, LL2, LL3, . . . , LLn, Maximum Height, Priority, Confidence (= 0 when first loaded),)

The Location points describe the base of the shadow or object.

In addition, if we know from intelligence sources where jammers are located we may enter them similarly as we have for discretes.

3 Pre-Flight Intelligent Filter Processor

These data represent antenna characteristics that will not change during flight, e.g. number of antenna elements and their configuration, antenna tilt angle and pointing direction, and location of the antenna on the A/C. It also contains the initial radar parameters, e.g. pulse repetition frequency (PRF), transmitter frequency, size of the data cube, and bandwidth of signal. The performance thresholds for evaluating antenna beam distortion are also initialized.

4 Pre-Flight Intelligent Detector Processor

These data represent data that are initialized but are not necessarily fixed, e.g. range resolution, Doppler resolution, top percentile for trim mean constant false alarm rate (TM-CFAR), and bottom percentile for TM-CFAR. Performance measure data are also set such as probability of false alarm thresholds for normal, low and very low levels of interest.

5 Pre-Flight Intelligent Tracker Processor

This state has similar data requirements as the pre-flight KB processor states [1-2]. All three processors have access to the same data. This state also sets the tracker processor performance measures and parameters, e.g. number of correct tracks, number of dropped or lost tracks, and kinematics of potential targets.

6 KB Performance Processor and Initiate System and Transient State.

The processor will monitor the AIRS queues for number of potential targets and registration of obstacles, discretes, clutter boundaries, shadow regions, and jammers.

7 KB Controller and Initiate System and Transient State.

The processor will initiate the antenna processing and monitor the system queues, auxiliary data correlations, feedback from the different processors, system errors, number of potential targets and registration of obstacles, discretes, clutter boundaries, shadow regions, and jammers.

8 Intelligent Filter Processor and Initiate System and Transient State.

Execute non-STAP algorithm, determine the secondary rings for each cell under test given the stored terrain features, run the non-homogeneous detector (NHD) algorithm if necessary, compute beam performance, and compute antenna weights based upon hypothesized KBC nulling tasks. Note for this state we don't want to distort the antenna beam pattern but gather data so the

KBC can determine if nulls should be placed in the direction of interferers and whether STAP is feasible.

9 Intelligent Detector Processor and Initiate System and Transient State.

The processor will implement thresholds as assigned, will default to the standard detection cell averaging algorithm, and use standard window sizes unless the cell of interest is at a clutter boundary.

10 Intelligent Tracker Processor and Initiate System and Transient State.

To initiate a track requires multiple CPIs. This process is just beginning. Correlations with objects and shadow regions have begun, performance measures are computed, (number of correct tracks, number of dropped tracks, and number of incorrect tracks) and tracks are formed. It reports tracks and potential correlations with other entities.

11 KB Performance Processor and Correlation, Assessment, and Learning State.

This processor will use the correlations obtained by the KB Controller [in state 12] for the first portion of its processing, i.e. until it has correlated or discounted all the discretes, clutter boundaries, road traffic, and shadow regions with a high degree of confidence. Once this task is completed the processor will insert synthetic targets of varying sizes and velocities to test the performance of the AIRS. During the second complete scan of an area the KB performance processor will be able to determine if the performance measures have improved. Based upon these results the performance processor may place targets in other locations and/or direct the controller where they should or should not use STAP.

12 KB Controller and Correlation, Assessment, and Learning State.

There are two levels of correlation required: 1. position of the above entities within a defined range ring and 2. the power level at the receiver given the distance to the entity. Note the definition of the range rings relative to the Earth contain different entities as the A/C moves. In addition, as the A/C moves different entities may require nulling, the AIRS may or may not want to place a null in their direction. Correlating entities by power may be done as defined by the following relations.

Road Traffic Power (Road ID #, Peak Power divided by average peak power over a defined window, for CPI #). Correlations are performed by a road object correlator algorithm using data from the detector and tracker processors. Power can be used to determine if the return signal varies differently from $1/R^4$ (where R is the target range) at the projected location of the road. If the majority of targets/tracks that originate from the location follow the road traffic pattern then their correlation is high.

Discrete's Power (Discrete ID #, Peak Power divided by average peak power over a defined range window, for CPI #). Correlations are performed by a discrete object correlator algorithm using data from the detector and tracker processors. Verify that the power varies as $1/R^4$ and the objects do not move, i.e. they do not generate a track.

Shadow/Obstacle Power (Shadow/Obstacle ID #, Peak Power divided by average peak power over a defined window, for CPI #). Correlations are performed by a shadow/obstacle object correlator algorithm using data from the detector and tracker processors. Correlating range ambiguity areas and dropped or coasted tracks help verify shadowed/obstacle locations. Shadowed regions loaded at pre-flight are computed from United States Geological Survey (USGS), or National Imagery and Mapping Agency (NIMA) databases, with an assumed flight path. If the databases are old then the terrain may have changed.

Data from IFF responses, outside sources, and other sensors are used to update jammer objects, aircraft, ground moving targets, and all unknown objects. Numerous data sources are used to register each CPI with ground "truth".

13 Intelligent Filter Processor and the Correlation, Assessment, and Learning Stage.

Rules as to when STAP should and should not be applied are required. It is assumed that the radar is flying in a known pattern and will be looking at the same scene each time it flies the same pattern. During the first complete flight over the defined scene the AIRS could execute a standard non STAP algorithm. The KB performance processor should place targets in non-homogeneous areas e.g. near roads and clutter boundaries. The position and type of synthetic targets are not made known to the KBC. In the second complete scan the KBC should attempt to use STAP where ever it can.

A method for determining if there is a sufficient number of training range rings for STAP is required. A method is to correlate each range ring with the terrain map to identify where there are discontinuities, major roads, etc. and label each region or sector-range with a terrain type. A classification code range ring correlator algorithm will implement this method in collaboration with the intelligent filter processor. The major or minor classification codes used in the USGS database, e.g. urban, forest, water, etc. will be used. Once range rings are chosen they can be evaluated for their homogeneity by using NHD. With a combination of the pre-flight loaded database, the use of the radar returns and the NHD, the system can "learn" which areas are homogeneous and evolve its rules as to which filter algorithms to employ.

During this state the controller will assign a low, medium, and high performance threshold levels for beam performance. This information along with requests of where to place nulls in the beam pattern will be provided to the intelligent processor. After a number of CPIs the KBC will evaluate performance measures from all the processors. Based upon this evaluation the KBC may assign different performance threshold levels and null requests for the filter processor.

14 Intelligent Detector Processor and the Correlation, Assessment, and Learning Stage.

This state uses the correlation data provided by the KBC to recognize terrain boundary locations. For those test cells within homogeneous regions the standard detection cell averaging algorithm and window sizes will be used. For those test cells near boundaries the CFAR processors will choose reference cells, algorithms, and window sizes as developed under the ES-CFAR program (1,5). The processor will perform detections, implement thresholds as assigned, re-compute and adjust probability of false alarm (Pfa) thresholds, evolve rules to apply the standard cell averaging rules, determine when to apply different algorithms, and when to recommend changing the detection threshold.

15 Intelligent Tracker Processor and the Correlation, Assessment, and Learning Stage.

Discrete objects, shadow regions, roads, and federal aviation administration (FAA) data will be obtained from the KB Controller and used to help correlate with existing targets and tracks.

Correlations of dropped tracks and highways will be performed with the KBC. Performance measures (number of correct tracks, number of dropped tracks, number of incorrect tracks) and sorting of tracks will be computed. It will report back to the KBC all its tracks and any discrepancies with the data obtained from the KBC. Discrepancies will be settled by the KBC and the other processors. As corrections are made the AIRS will evolve its rules and learn.

16 KB Performance Processor and the Steady State.

The performance processor will constantly measure the performance of all processors to determine whether AIRS is performing better. The processor will continually look for changes or requests submitted by the user or changes in data from outside sources. It will monitor performance by checking the beam pattern performance data, detection data, and track data. It will insert known radar cross section (RCS) synthetic targets at locations where there are boundaries in terrain types and evaluate the detection capability of the system. By placing different targets at different locations the performance of the current rules can be computed. If performance is low then the rules being used by the KBC will be modified.

17 KB Controller and the Steady State.

The KBC will access the same performance measures as presented in [16]. Based upon these performance values the KBC will asses its current rules and apply changes accordingly. The rules the KBC can change are based upon a processor's reported data and the user requests, such as change in the antenna's beam pattern and the A/Cs flight path.

18 Intelligent Filter Processor and the Steady State.

This processor will monitor its beam pattern performance. It will change its rules based upon the environment and the number of nearby jammers and discretes. For example, the processor should manage the number of degrees of freedom required to notch jammers and descretes and yet maintain enough degrees of freedom to perform STAP processing. It will measure its own performance and report it to the KBC for total sensor performance evaluation.

19 Intelligent Detector Processor and the Steady State.

During this state its processor measures performance based upon the number of detections and number of false alarms. It will increase or decrease the threshold level, change window sizes for CFAR algorithms, and change rules for choosing CFAR algorithms based upon previous flights over the same or similar clutter interfaces.

20 Intelligent Tracker Processor and the Steady State.

This state measures performance based upon number of correct tracks, missed tracks, and number of false tracks. Based upon these numbers and the terrain, the processor will adjust its rules and thresholds to increase its performance.

Unmanned Air Vehicles

Portions of AIRS to date have been applied to an airborne radar surveillance system flying a repetitive route accumulating data and knowledge to be used during the next sortie. For example the knowledge that can be gained, are the correlation of road traffic with moving targets, the location of shadowed regions by mountains, or the location of large discrete targets and tunnels. This knowledge can subsequently be used for changing the algorithms and providing information throughout the signal and data processing chain. However, today's adversaries are not traveling in truck convoys, flying aircraft in formation, or traveling the desert in tanks. They cannot easily be detected and tracked with stand-off airborne sensors such as AWACS or JSTARS. Today's adversaries are embedded in urban environments traveling in everyday vehicles, without uniforms, and carrying small weapons and bombs. Large surveillance platforms cannot easily detect and track individuals with weapons, trucks, automobiles, or weapon caches housed in dense urban areas or mountainous regions that are kilometers away. To meet these requirements, organizations are investigating unmanned air vehicles (UAV) with different sensors which can be deployed in urban and mountainous regions to detect and track various targets. These UAVs may operate on their own, in conjunction with surveillance platforms, or with minimum human intervention. This section addresses some of the issues with applying an AIRS architecture to sensors aboard UAVs.

There are numerous research efforts being sponsored in the US regarding UAVs and unmanned robotic vehicles (URVs). The USAF has a program called MultiUAV [6], a simulation model, based upon a series of managers that cooperate in the managing of multiple UAVs. These managers or agents perform tactical maneuvering, as well as sensor, target, cooperation, route and weapons management. The sensor agent maintains a list of detections, performs automatic target recognition, and also performs battle damage assessment. The simulation is built in Mathwork's Simulink software and allows for a set of UAVs to perform a mission together where each is assigned a predetermined search pattern. When a target is detected, all UAVs are notified and, since they have identical software, they re-plan the same or next set of tasks. UAVs communicate over two data buses, one for simulation related signals and one for the actual communications that would go on between UAVs.

The US Navy is pursing a program called Autonomous Intelligent Networks and Systems (AINS) [7] that involves UAVs that can cooperatively work together to achieve a goal without human intervention. The program has three goals or focus areas: exploring intelligent autonomy, providing wireless communications between elements without a given infrastructure, and providing intelligent control of the elements by humans. Another underlying goal is to have their network of elements not be dependent on a global positioning system (GPS). All sensors are passive, e.g. no monostatic radar systems. The program has demonstrated nine helicopters flying safely in a limited space.

DARPA is funding multiple efforts related to UAVs. One area involves emulating how ants, via the use of pheromones, communicate and gather food. There are three ways ants direct their fellow ants to food [8]. They lay pheromones for attracting many ants to the food, certain types of ants will return to the nest and vibrate its antenna to attract another ant to acquire food, or they may vibrate their antenna differently to attract numerous ants to accomplish the same. Researchers have been studying pheromone usage to model how UAVs can work autonomously. Some researchers emulate the deposition and interaction with pheromones by the environment and its walkers [9]. The environment is partitioned into numerous shaped areas where each area receives pheromones placed by the walkers. The areas aggregate the amount of pheromones placed by the walkers and evaporate pheromones over time. The areas also diffuse the pheromones to nearby areas in the environment. It is through the walkers placing and "sensing" the pheromones that communications exist between the walkers or UAVs. Different flavors of pheromones may convey different features such as the detection of hostile versus friendly entities and they have different semantics. Different flavors with the same semantics for example may have different evaporation rates, propagation rates or different thresholds.

Digital pheromone technology was demonstrated by Johns Hopkins University and the US Army with four robots controlled by algorithms within a mock urban area along with two UAVs also controlled by pheromone technology. According to [10] "The actions of the vehicles were not scripted as evidenced by their adapting to the unplanned failure of one of the ground robots. Rather than specify each vehicle's task, the operator simply gives a high level command to the whole swarm, such as 'survey this area and track any identified targets' or 'patrol around this convoy'. The robots autonomously configured themselves to determine which robot would perform what task in order to accomplish the overall objective. The operator was free to monitor their behavior, receive their reports, and provide additional guidance as needed when priorities or mission objectives changed. The swarm did not need any special configuration to meet a wide variety of mission requirements, respective of the operating environment or the number and type of vehicles involved."

The above referenced research is just a sample of the work that is currently being pursued to guide autonomous vehicles on the earth and in the air. According to [10] an industry survey has identified 48 research and development programs using emerging methods for future advanced flight control concepts. Some researchers are studying the use of genetic algorithms in order to breed control behaviors. DARPA's Software Enabled Control (SEC) program is deliberately avoiding using genetic algorithms. They are researching algorithms [10] that "generate behaviors that are possible as a group but not possible as individuals." The program believes that the passive sensing of the kinematics of its neighbor is needed in coordinating collision free flight paths. They also believe that the UAVs must respond to human control and never perform any unexpected actions.

There are technical issues associated with applying the AIRS architecture to UAVs. The designs shown in figures 1 and 2 are viable for a UAV as long as the technology employed can meet the size, weight, and power requirements of the UAV. Semantic Web technologies for communicating and controlling the various sensors aboard a UAV and between UAVs are required. The manner in which sensors communicate will change depending upon the deployment and communication media. If we deploy UAVs in urban environments, then some preloaded data and information will be different, i.e. building locations, their size, construction, etc.

In the current AIRS architecture the radar is assumed to be flying aboard a surveillance aircraft viewing the earth looking for multiple threat targets. This deployment makes use of map data, outside information sources, and due to the repetitive race track route that is traversed, it can "learn" from the experience and modify its radar signal processing algorithms. The learning process of monostatic or multistatic radars mounted on UAVs, which do not travel repetitive routes, requires investigation. We must address how UAV radars learn from the terrain, outside sources, effectively manage its resources, and communicate between other sensors and information sources. As an illustration consider a digital implementation of a pheromone paradigm for managing UAVs. For example, an urban environment would be partitioned into areas where UAVs can read and write information (i.e. sense and leave pheromones) to a ground node, the ground nodes can be integrated together and provide information to the GIG complying with the Net-Centric paradigm. New learning algorithms need to be developed to use the knowledge acquired by pheromone model communications between sensors. For example, updates to map data can be left by one UAV to be read by others, detections and images can be shared via the pheromone model, and then allow the UAVs to work together to learn what previous sensors have discovered. The nodes within each area will contain and maintain the knowledge for each new UAV sensor platform that visits the area.

KB Tracking

In the previous sections we presented an overview of AIRS and its end-to-end processing sequence. We discussed its feedback and learning structure and the sharing of information and data from outside sources. In two other papers contained herein Dr. Wicks provided information about expert system Constant False Alarm Rate (CFAR) processing and the results of a study where the choice of secondary or training data for STAP filtering was greatly enhanced by using map data. These are two examples of using external knowledge and KB processing for enhancing the performance of radar signal processing. The rules for picking the best training rings or the best CFAR processing algorithms are both based upon knowledge of the terrain obtained from map data. The rules for their choice were hypothesized by the researchers and then tested by using actual radar data. This same procedure is recommended for the development of AIRS.

To complete the end-to-end processing architecture of an AIRS we will present a KB tracking algorithm that extends our US Air Force (USAF) funded work (2). We will present an overview of this tracking algorithm and some of its AI rules e.g. maneuver or obstacle rules and shadow rules. An AI logic structure for implementing these rules is discussed next and some additional rules for our AIRS design are provided.

The logic structure is independent of any tracking algorithm and can address aircraft or ground moving targets. It is compatible with the overall AIRS design and is modifiable. The thrust of this logic structure is to utilize as much auxiliary data (e.g. maps, other sensors, target kinematics, and radar platform characteristics) as possible to maintain individual identifiable tracks. With today's tracking algorithms if a track is dropped and another track is formed there is minimum effort expended to determine if the two tracks were formed from the same target. If a track is dropped algorithms, for the most part, do not investigate why and then use this information in enhancing the overall signal processing performance. Algorithms do not learn based upon their previous performances. They are memoryless once a track is dropped. The proposed logic structure presented herein addresses these issues and investigates the potential for building an AI based tracking algorithm.

Our current tracking algorithm (2) has three separate instantiations. There is an uncoupled two state alpha beta filter with position and velocity component states, an uncoupled three state Kalman filter with position, velocity, and acceleration component states, and an extended four state Kalman filter with both x and y position and velocity component states. The tracker gathers reports, evaluates the reports and correlates them with known tracks, forms a correlation matrix and distance matrix, performs an association logic based upon nearest neighbor and oldest track, and performs track maintenance i.e. update extant track states, spawn new tentative tracks with unused reports and drops tracks with a state value of zero. A diagram illustrating the state logic is shown in figure 5. A new tentative track is given a state of 1. If its projected position is detected again on the next coherent processing interval (CPI) it is given a state of 2, and so on. Once the target is in state 4 it is considered in a firm state as long as it is still detected for each subsequent CPI. Once in the firm state, if there are four consecutive CPIs in which the target is not detected (i.e. a Miss) then the track is dropped. It is our contention that once a tentative track exists then we should maintain its history even if it receives one or more misses. This is important in order to correlate false or dropped tracks with roads, or jammers, discretes, shadow regions, etc. This information is needed to feed back to the KBC and to the other processors as discussed in the previous sections.

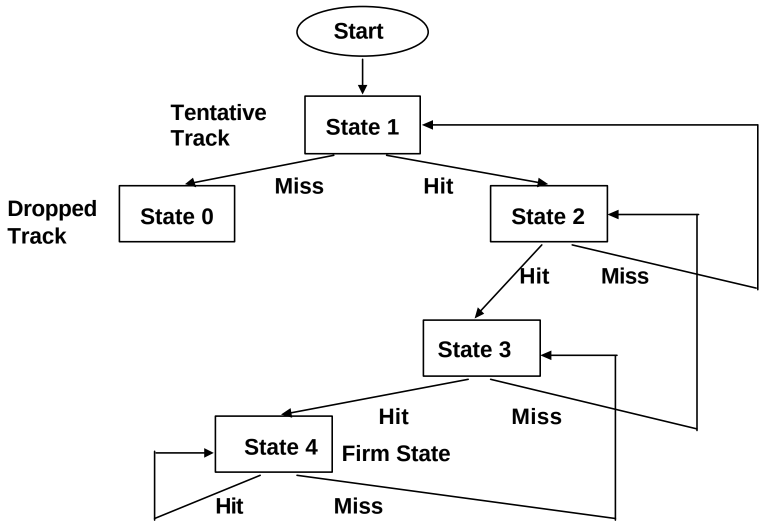

Figure 5. Integrating AI Rules

The following is a preliminary design of a logical structure to capture AI rules for the tracking portion of AIRS. It is by no means complete and does not address each of the numerous attributes for tracking any specific type of target (e.g. aircraft, ground vehicles, missiles) for all its possible scenarios embedded in all possible environments or clutter. It is constructed to work with a radar tracking filter such as alpha beta or Kalman. The logical structure is shown in figure 6. It is an abstract model and will require numerous detail level designs before it can be coded and tested. The logic is described using alpha characters to indicate where in the structure we are referring. Throughout the description the use of outside data sources is illustrated and the addition or verification of data sources is presented.

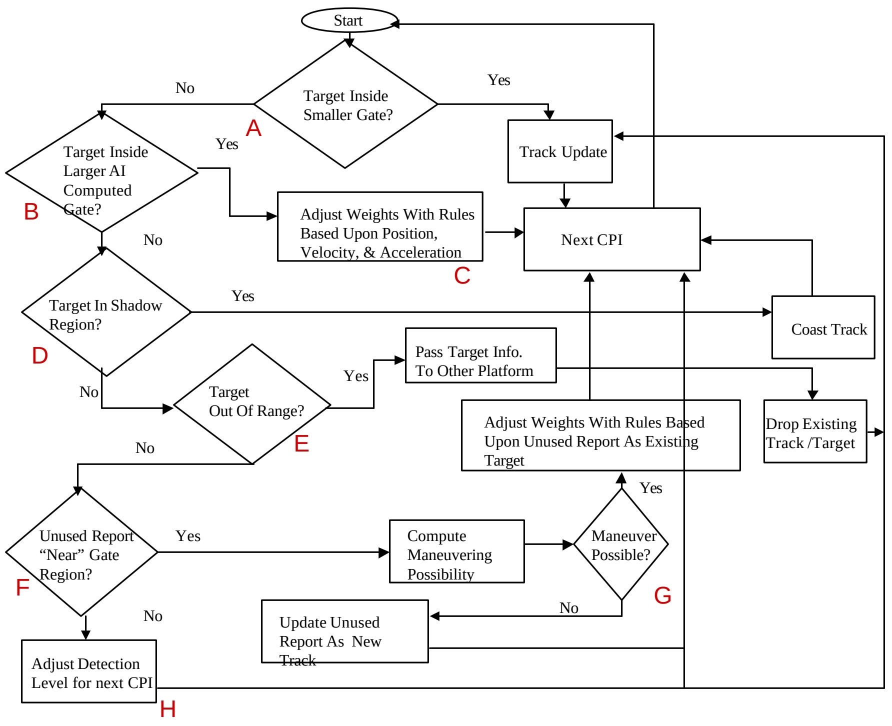
Start
Target Inside Smaller Gate?
A
Yes
No
Track Update
Target Inside Larger AI Computed Gate?
B
Yes
Adjust Weights With Rules Based Upon Position, Velocity, & Acceleration
C
Next CPI
No
Target In Shadow Region?
D
Yes
Coast Track
Pass Target Info. To Other Platform
Yes
No
Target Out Of Range?
E
Adjust Weights With Rules Based Upon Unused Report As Existing Target
Drop Existing Track /Target
No
Yes
Unused Report "Near" Gate Region?
F
Yes
Compute Maneuvering Possibility
Maneuver Possible?
G
No
Update Unused Report As New Track
No
Adjust Detection Level for next CPI
H

Figure 6. Logical Structure

A. Within this decision block (A) we are asking whether a detected target is within the gate of a known and therefore projected track. If the answer is yes then we simply update the track using the tracking filter of choice (e.g. Kalman). If however a target is detected and it is not within any projected track's gate (i.e. an unused report) then we need to determine whether it lies in a larger AI computed gate. The idea of using more than one size or variable size gate is discussed in the literature. Skolnik (3) suggests "The size of the small gate would be determined by the accuracy of the track. When a target does not appear in the small gate, a larger gate would be used whose search area is determined by the maximum acceleration expected of the target during turns." Brookner (4) states while discussing the g-h filter

"However, aircraft targets generally go in straight lines, rarely doing a maneuver. Hence, what one would like to do is use a Kalman filter when the target maneuvers, which is rarely, and to use a simple constant g-h filter when the target is not maneuvering. This can be done if a means is provided for detecting when a target is maneuvering. In the literature this has been done by noting the tracking-filter residual error, that is, the difference between the target predicted position and the measured position on the nth observation. The detection of the presence of a maneuver could be based either on the last residual error or some function of the last m residual errors. An alternative approach is to switch when a maneuver is detected from a steady-state g-h filter with modest or low g and h values to a g-h filter with high g and h values, similar for track initiation. This type of approach was employed by Lincoln Laboratory for its netted ground surveillance radar system. They used two prediction windows to detect a target maneuver. If the target was detected in the smaller window, then it was assumed that the target had not maneuvered and the values of g and h used were kept ... If the target fell outside of this smaller 3 sigma window but inside the larger window called the maneuver window, the target was assumed to have maneuvered."

B. These references were provided to indicate that the radar community has tried different approaches for varying the gate sizes for tracking maneuvering targets. The Kalman filter is more suited for maneuvering targets. However, a universal method for choosing a larger gate size because of a maneuver is not well established. If the larger gate is too large then multiple targets may occur within them. The maneuverability of a target is target dependent and may be human dependent and very unpredictable. What we are proposing is that the larger gate be built using AI techniques. Let the history of the target's flight and a priori knowledge about a potential target dictate how to compute the larger AI gate.

Since we are building an intelligent surveillance system we will have data obtained from sources outside our radar system, e.g. map data, intelligence data, and other sensors. We can assume we know what type of targets we are tracking, such as helicopters, tanks, scud launchers, surveillance aircraft, fighter aircraft, and missiles. If so then we know something about their kinematics, i.e. their minimum, maximum and average velocities for different altitudes, their maximum gravitational (G) force turn they can withstand and at what radius, and their maximum acceleration. Using these data we can construct rules that will compute the larger size gate based upon a degree of belief given the type of target, e.g. helicopter or a fighter aircraft. This degree of belief can be computed using information from outside data sources, its previous kinematics data (velocity, location, etc.), radar cross section, and altitude amongst other factors such as the type of mission, its position in the scene, and sensitive locations or targets.

A simple rule is to take the maximum velocity for the target type that has the highest belief and compute the maximum distance it could have traveled from the previous position on the last CPI. This allows us to compute a semi-circle around the vector the target was heading. See figure 7. This approach may be fine for a target like a surveillance aircraft, but not for a tank or track vehicle or scud launcher. For example, a tank which can easily turn 180 degrees, a circle may have to be drawn with radius equal to the maximum distance that can be traveled within the time between CPIs. The more we know about the targets we are tracking the more intelligent we can be in designing our rules and estimate our gate sizes.

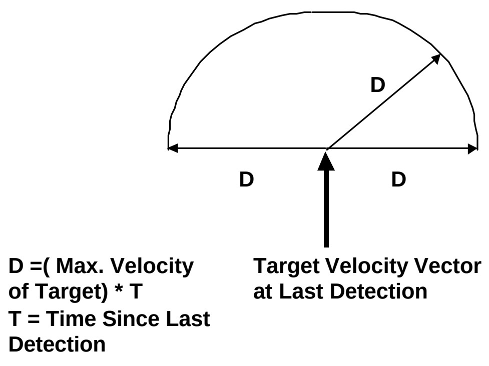
D
D
D
D =(Max. Velocity of Target) * T
T = Time Since Last Detection
Target Velocity Vector at Last Detection

Figure 7. Example AI Computed Gate

C. If the target is detected in the larger gate then we need to adjust the weights of our tracking filter. Indicated in block C we can adjust the weights with rules based upon position, velocity and acceleration. These rules can be simple, e.g. if the target was detected in the larger gate then set the weights for the next CPI as if the target were detected the first time. This will eliminate any memory or smoothing that the filter had performed and start off with a larger gate size. More sophisticated rules can be employed and should be investigated further, dependent upon the tracking filter used.

D. If the target was not found in the smaller or the larger gate then we need to determine if it is being shadowed from our radar, possibly by terrain. Our logic is assuming that the radar system has a priori data that are available such as terrain data containing elevation attributes, roads, and bridges. With this information we can compute whether or not given the elevation of the radar and the last position of the track if there is terrain obstructing the radar's illumination of the target. If there is an obstruction then we should be able to project, given the last known velocity of the track and the changing position of the radar, how many CPIs the track will be obstructed. Based upon these computations we can then coast the track until the next CPI. For each coasted CPI we should also look for new unused reports that can occur due to our coasted track changing its projected velocity while it is being obscured. See figure 8. If this does occur and a new track is initiated we should "flag" this track that it may be the coasted track. Once we compute when or which CPI the original track should be visible and if it isn't, even after two additional CPIs, we should then revisit the new track. During this revisit we need to compute whether or not the dynamics of the target/track were capable of maneuvering to the position that the radar detected the target. (See paragraphs F and G for more details.) If it is shown possible, then the new track should be updated as being the old track with some degree of belief. If however, the original track is detected after it has moved beyond the obstruction then we should go back to the new track that was initiated and remove the "flag" indicating the possibility that this was a firm track that was coasted.

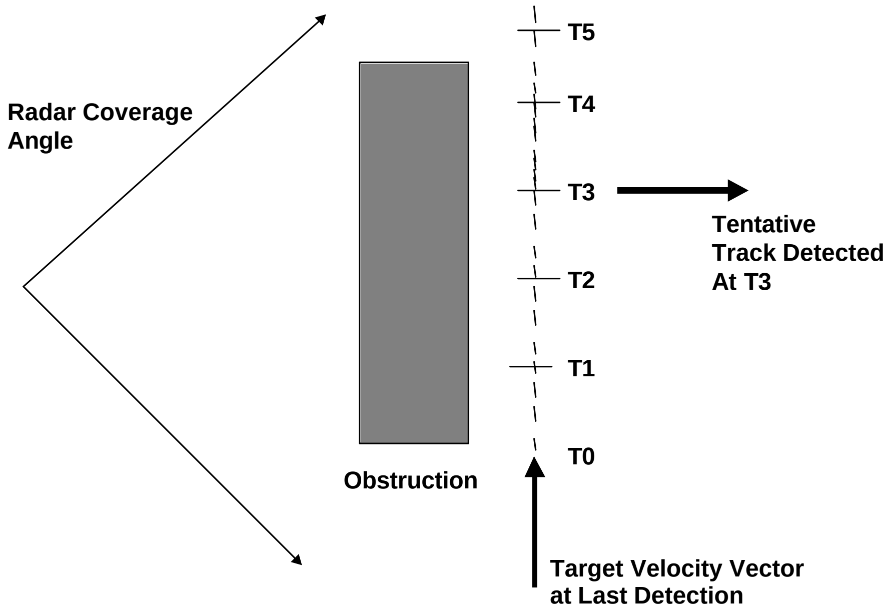
Radar Coverage Angle
Obstruction
T5
T4
T3
T2
T1
T0
Tentative Track Detected At T3
Target Velocity Vector at Last Detection

Figure 8. Track Obstruction

E. If the target is not in either gate and it is not shadowed then maybe the target is out of range. This is easy to compute given its last position relative to the radar. If it is out of range then we should pass this information to another sensor platform along with the track data we have acquired. The knowledge of when a target is going to reach this point can be predicted earlier than the last CPI. However, the point in space when a target will be out of range is a variable dependent upon the radar and the target’s movements.

The information that can be passed to the other platform can contain the time of the first acquisition, its history path, velocity range, hypothesis of type of target, and any other kinematics or knowledge that has been gathered throughout its track. This data can be used by the message receiving platform in assigning degrees of belief about the target's maneuverability, type of target, and identification.

F. If the target is not in either gate, not shadowed, and not out of range then what happened to it? Maybe our knowledge about its kinematics was incorrect? Maybe our sensor and filtering model has more error variation than we thought? Maybe the target maneuvered and its radar cross section (RCS) is too low and therefore not detected. Maybe the clutter is too large and we can't detect the target?

What we can do is determine if there are any unused reports. If unused reports exist then maybe one of these are our target. First we need to perform a quick culling to determine if at maximum velocity (Vmax) our target could have traveled from where we last detected it to where the unused report was detected, a distance of D. If Vmax times T (time between the two detections) is less than D then this unused report can't possibly be due to the same track. If all unused reports result in the same finding then we conclude that there are no unused reports that may be due to our track. If however, one or more computations show that the distance to the possible reports could have been traveled by the target then we need to compute its possibility and assign a degree of belief to each report.

G. A simple algorithm for computing the possibility of an A/C maneuverability is illustrated in figure 9. D is the distance between the last detection and the position of an unused report. The different radii (R1 and R2) represent the different radius that one can construct that can pass a circle or arc through the two end points of the chord of length D. If we assume that the acceleration is a maximum then we can assume that the velocity is our last estimated velocity or its maximum velocity. Each assumption has a certain amount of error. We can compute different values of R by the following:

Rest = $(\text{Vlast})^2$/Accmax,

Rmax = $(\text{Vmax})^2$/Accmax.

For different values of R and D we can compute the distance of the arc connecting the end points of the chord D. It can be shown from figure 9 that:

Theta = 2(arcsin((D/2)/Rest)) or
Theta = 2(arcsin((D/2)/Rmax)).

The distance along the arc is 2*Pi*Rest/(Theta/360) = Darcest. Therefore if at (Vlast)*T is less than Darcest then the maneuver is not possible. Similarly if (Vmax)*T is less than Darcmax then the maneuver is not possible.

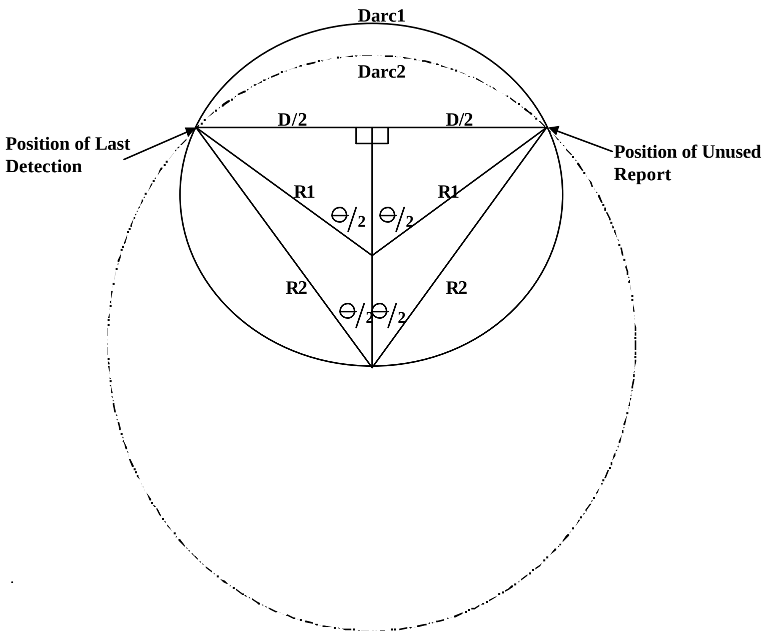

Figure 9. Maneuver Possibilities

Similar rules can be developed for different targets and their kinematics to determine the best rules for each. The developed rules can be verified and modified by consulting with experts who are aware of a target's kinematics.

H. If the target is not in either gate, not shadowed, not out of range, and our kinematics is verified then what happened to the target? It may have maneuvered such that its RCS decreased. If it's a ground slow moving target it may have stopped. It may be hidden by a tunnel. The level of detail for examining why a target track is undetectable needs to be perused dependent upon the target, the environment, the amount of detail a priori data available, and the scenario under investigation. For this iteration of our AI logic structure we have elected to halt our level of investigation and to coast the target. The algorithm would request the KBC to reduce the detection level for the location which we lost the target and the locations where we project the track to be for the next four CPIs. We should identify that the track is potentially dropped and treat the track as a coasted track. If after four CPIs it cannot be correlated with a detection then the tracking filter will drop the track.

Summary

This paper has provided an overview of a hypothesized integrated end-to-end radar signal and data processing chain. The paper has discussed how the use of ontologies can be used for sensors to communicate and share information on board the same platform and between platforms. The majority of the paper was devoted to describing an airborne intelligent radar system (AIRS). A description of the AIRS architecture was provided along with a detailed and high level description of the four states of processing and the functions performed by the KB performance processor, KB controller and the filter, detector, and tracker processors. This section was followed by information related to the implications of porting AIRS to sensors mounted on UAVs. The last section briefly described a tracking algorithm and proposed an AI logic structure for incorporating rules for different targets, environments, and scenarios. The driving force of this logic structure is to use AI to learn about each track and to analyze each track completely before it is dropped. The logic structure is independent of any tracking algorithm, environment, target type, or scenario.

The AIRS architecture is new and revolutionary. Its potential is great. It is one element in a bigger program dealing with waveform diversity and sensors as robots.

Acknowledgements

The authors would like to recognize the efforts of the following people. We would like to thank Mr. Gerard Genello, Mr. William Baldygo, and Dr. Mike Wicks for providing the resources, encouragement, guidance, and the opportunity in the pursuit of our goals. We would also like to thank Mr. John Spina for chairing this lecture series and Mr. Christopher Capraro and Mr. Gerald Berdan of Capraro Technologies, Inc. for their help throughout. Thank you all.

References

[1] W. Baldygo, M. Wicks, R. Brown, P. Antonik, G. Capraro, and L. Hennington, "Artificial intelligence applications to constant false alarm rate (CFAR) processing", Proceedings of the IEEE 1993 National Radar Conference, Boston, MA, April 1993.

[2] R. Senn, “Knowledge Base Applications To Adaptive Space-Time Processing”, Unpublished Final Report, July 1999.

[3] M., L., Skolnik, "Introduction to Radar Systems", McGraw Hill, New York, 1980.

[4] E., Brookner, "Tracking and Kalman Filtering Made Easy", Wiley, New York, 1998.

[5] M. C. Wicks, W. Baldygo, and R. D. Brown, “US Patent 5,499,030 Expert System Constant False Alarm Rate (CFAR) Processor”, filed March 18, 1994 issued March 12, 1996

[6] Rasmussen, S. J. and Chandler, P. R., “MultiUAV: A Multiple UAV Simulation For Investigation of Cooperative Control, Proceedings of the 2002 Winter Simulation Conference.

[7] Lawlor, M., “Miniaturization, Networking Pervade Future Unmanned Systems”, SIGNAL Magazine, April 2003.

[8] Bonabeau, E and Meyer, C., “Swarm Intelligence: A Whole New Way to Think About Business”, Harvard Business Review, May 2001

[9] Parunak, H. V. D., Brueckner, S., and Sauter, J., ”Synthetic Pheromone Mechanisms for Coordination of Unmanned Vehicles”, AAMAS 2002, Bologna, Italy

[10] Adams, C, “Aviation Today UAVs That Swarm”, Avionics Magazine, July 15, 2005