AFRL-IF-RS-TR-2003-20
Final Technical Report
February 2003

AUTONOMOUS NEGOTIATING TEAMS (ANTS) ADAPTIVE RESOURCE MANAGEMENT FOR ELECTRONIC WARFARE

BAE Systems

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. H356

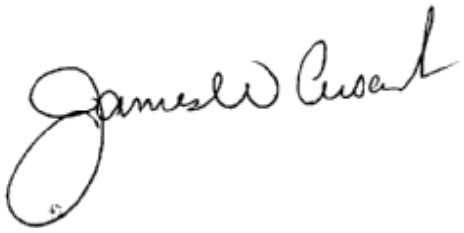
AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-20 has been reviewed and is approved for publication

APPROVED: 

EDWARD L. DEPALMA
Project Engineer

FOR THE DIRECTOR: 

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	FEBRUARY 2003	Final Aug 99 – Aug 02

4. TITLE AND SUBTITLE
AUTONOMOUS NEGOTIATING TEAMS (ANTS) ADAPTIVE RESOURCE MANAGEMENT FOR ELECTRONIC WARFARE

5. FUNDING NUMBERS
C - F30602-99-C-0167
PE - 62301E
PR - H356
TA - 01
WU - 01

6. AUTHOR(S)
Paul D. Zemany

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BAE Systems
PO Box 868
MER15-2403
Nashua New Hampshire 03061-0868

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFSF
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2003-20

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Edward L. DePalma/IFSF/(315) 330-3069/ Edward.DePalma@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The Autonomous Negotiating Teams (ANTS) program objective is to develop and demonstrate a Dynamic Scan Scheduler (DSS). Under the ANTS program, BAE built a testbed for DSS development and provided EW domain-related information to Northeastern University (NEU) and SRI. Both NEU and SRI then developed DSS approaches. These approaches were compared to the Fixed Scan Scheduler (FSS) using the testbed. Metrics relating to mission success was used to show the benefits of the DSS over a FSS. Under the ANTS Challenge Problem, BAE had the task to provide a hardware/software testbed to the other ANTS investigators. Since the ANTS program addresses "time critical resource management of distributed resources", the testbed had to involve distributed hardware that requires coordination using limited bandwidth communication. To accomplish this, a three beam CW doppler radar sensor was built and the objective was to use four (or more) of these "sensor nodes" to detect and track a moving target. Under this Challenge Problem task, BAE designed and built several testbed sets and provided them to the other ANTS investigators. Software that supports the testbed was also provided and a test benchmark was conducted using four nodes to demonstrate tracking.

14. SUBJECT TERMS
Dynamic Scan Scheduler, Fixed Scan Scheduler, Time Critical Resource Management, Autonomous Negotiating Teams, Agent Technology

15. NUMBER OF PAGES
96

16. PRICE CODE

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

Table of Contents

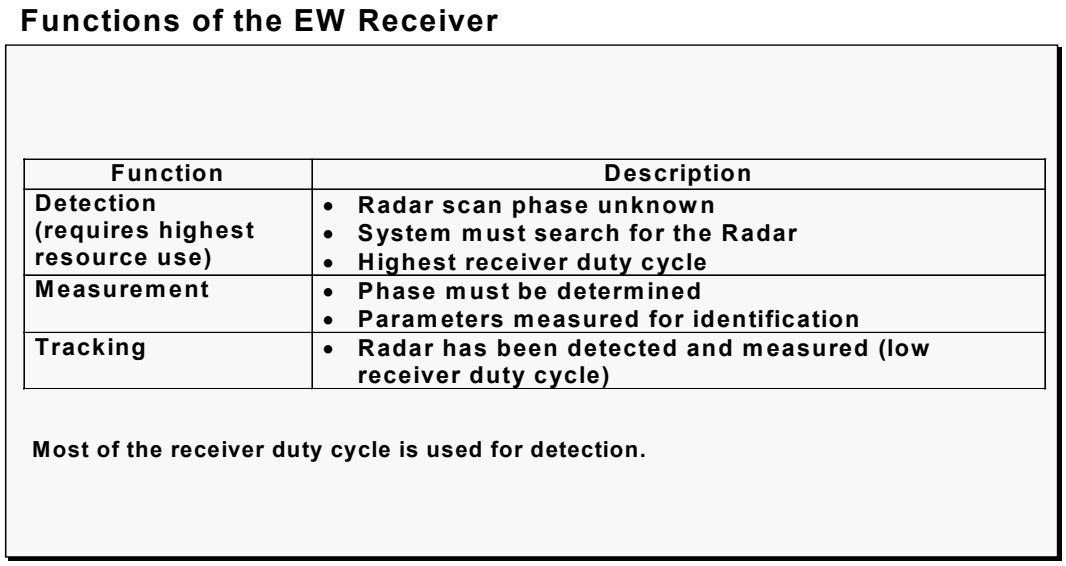
List of Figures

List Of Tables

ANTS PROGRAM OVERVIEW

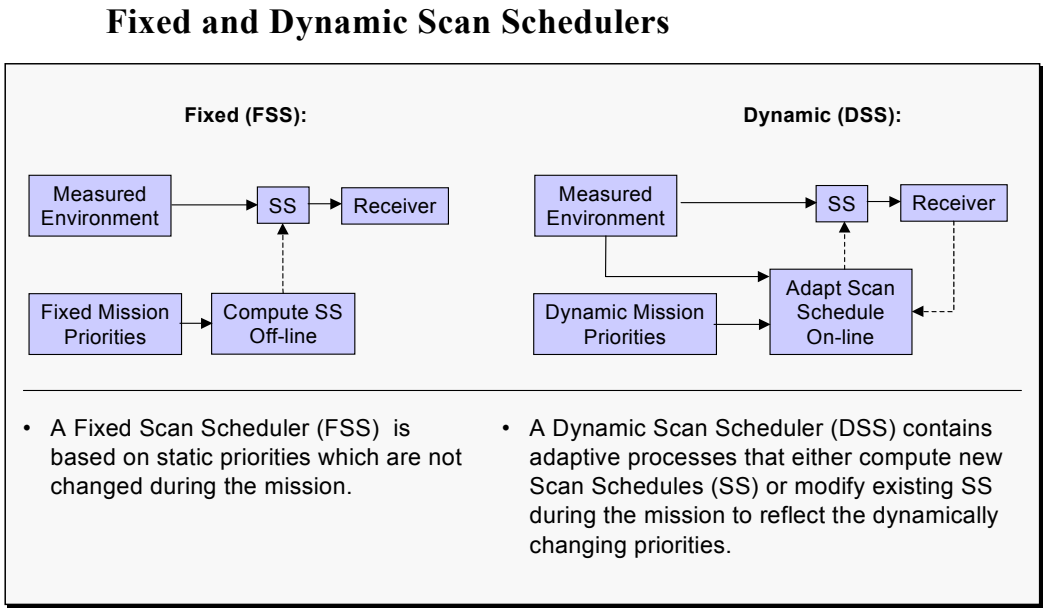
The ANTS program conducted by BAE and subcontractors Northeastern University and SRI focused on technology for time critical resource management. The approaches were applied to electronic warfare (EW). In EW, one of the tasks is to control the EW receivers in order to optimize the detection, measurement, and tracking of threat radars. Figure 1 describes the three main functions of the EW receiver.

FIGURE 1 Functions of the EW Receiver



A resource management process to generate a "scan schedule" (SS) that controls the EW receiver. Traditional approaches involve producing the SS offline in advance of the mission. This is referred to as a fixed scan schedule (FSS). Information on the threat radars and mission goals is used to develop the FSS. However, the mission can have unexpected events which change goals and expectations involving the threat environment. Therefore a new SS, based on the new situation, should be used. A dynamic scan schedule (DSS) is needed to respond at the mission takes place. Figure 2 shows the differences between a fixed scan schedule (FSS) and a dynamic scan schedule (DSS).

FIGURE 2 Fixed and Dynamic Scan Schedulers



The objective of the ANTS program is to develop and demonstrate a DSS. Under the ANTS program BAE built a testbed for DSS development and provided EW domain related information to NEU and SRI. Both NEU and SRI then developed DSS approaches. These approaches were compared to the FSS using the testbed. Metrics relating to mission success was used to show the benefits of the DSS over a FSS. The first part of this report describes the approaches and results obtained in the scan schedule work.

The other main task under the ANTS program was the "challenge problem". BAE had the task to provide a hardware/software testbed to the other ANTS investigators. The testbed was required to be low cost so that the ANTS investigators could obtain a copy. In addition, since the ANTS program addresses "time critical resource management of distributed resources", the testbed had to involve distributed hardware that requires coordination using limited bandwidth communication. To accomplish this, a three beam CW doppler radar sensor was built and the objective was to use four (or more) of these "sensor nodes" to detect and track a moving target.

Under the Challenge Problem task, BAE designed and built several testbed sets and provided them to the other ANTS investigators. Software that supports the testbed was also provided and a test benchmark was conducted using four nodes to demonstrate tracking.

The results of the CP are important because they provide real constraints to the ANTS technology developers. In addition, the solutions developed will be useful in future systems involving distributed "micro" sensors. The second part of this report discusses the challenge problem.

SCAN SCHEDULE OVERVIEW

The objective of this ANTS scan schedule effort was to design, develop and demonstrate an Autonomous Negotiating Teams (ANT)-based dynamic scan scheduler to improve existing fixed sensor allocation, with an adaptive scan scheduler responding to the evolving threat environment, based on adaptive software architecture hierarchies and distributed ANT technologies. Improvement was measured through increases in probability of rapid detection of threat radars. Demonstrations of the DSS were conducted. Two approaches were investigated and compared to a "fixed" scan schedule approach.

Dynamic scan schedule (DSS) and Distributed DSS (DDSS) characterization

DSS and event responsive scan scheduling is a need common to several EW programs. Coordinated inter-platform scan schedules will be a technique for handling both dense and sporadic threat environments. The primary need is for independent though coordinated scan schedules each executing independently on separate platforms. Improvements are expected from the DSS alone as well as the coordination made possible by the DSS. A DSS approach makes it possible to implement a "Distributed Dynamic Scan Schedule" (DDSS). A DDSS requires communication between platforms. This communication via a bandwidth-constrained link that can be characterized by a maximum, minimum and typical long-term bandwidth and also by a burst bandwidth capability. The link will also have a typical latency that varies between a max and min. Latency may be dependent on message priority, with only short messages able to receive the highest priority. Latency is likely to be driven by other activities taking place in the environment at the time.

Each platform's scan schedule will operate independently since each will see a different (even if only slightly different) signal environment because each is in a different location, orientation, and is looking at a different spot in space. The scan schedule must schedule a number of fixed assets for looks or dwells in space, time, and frequency.

.

Scan Schedule Testbed

The goal of the SS testbed is to run the ANTS scan schedule in real or near real time. Real time threat simulators are available at BAE, but they cost upward of $20 -30M. They are expensive to program and labor intensive to use. For this reason, ANTS used a reasonable compromise, that exploits some of the same decisions, compromises, and hardware approaches take by main line tactical EW programs. The ANTS testbed emulates the scenario at the dwell period level instead of the RF pulse level. This is done because the ANTS controlled scan scheduler schedules receiver dwells that must

intercept several RF pulses. Thus simulation at the dwell time scale is suitable for ANTS simulation.

Both high fidelity classified or unclassified pseudo threat definitions can be generated. The real time nature of the scan schedule is exercised by FPGA hardware based adaptive window filters that emulate the spatial frequency and time windowing of a convention tactical EW receiver.

The test bed was used to develop and test various scan schedule approaches. To determine the effectiveness of the DSS, a FSS was compared to a dynamic scan schedule (DSS) using scenario with typical emitters. The most useful metric was the number of radar dwells before detection.

Two approaches for the DSS were considered. The SRI "on the fly" generation approach used a rapid SS generation to produce a FSS in response to changes in emitter detection priorities. This allows a new FSS to be produced in real rime during the mission. The NEU control based approach uses feedback from the detection process to adjust the SS. This adjustment allows emitters to be monitored after they are detected. This monitoring process is referred to as "tracking".

Both the SRI and NEU DSS approaches were tested using the BAE test bed. Detailed results as well as a detailed description of the approaches are presented in this report.

Scan Schedule Requirements

BAE provided a requirements specification for a DSS. This section presents the background information needed to develop DSS approaches.

The objective of the scan schedule is to optimize the use of the EW receiver for detection, monitoring and measurement of threat radars. The SS process uses a-priori information as well as data collected during the mission to control the receiver.

Detection places the greatest load on the EW receiver use because there is a wide range of frequencies to monitor and the states of the radars to be detected are unknown. For detection, it is possible (and desirable) to determine the scan schedule in advance. Conditions that require a scan schedule change are when the set of radars or priorities change or the receiver is needed for monitoring or measurement or other platforms are able to help. In these cases, a new scan schedule is needed. The scan schedule is set up to optimize the detection process. This setup process can be done prior to the mission by an offline process. During the mission, the mission processor can switch the scan schedule or even recompute a new scan schedule. The "detection" scan schedule can change in response to the following:

Information obtained during the mission changes the detection priorities of radars

Information obtained during the mission identifies a new set radars to be detected.

A cooperating platform offers to share the detection load.

As a result of detection, the receiver resource is needed to conduct additional measurements.

The receiver is needed to monitor a detected emitter.

To address 1-3 above, an online process takes the set of radars of interest and computes a table of dwells (i.e. a fixed scan schedule) The ability to generate new fixed scan schedules during a mission represents an improvement to current approaches.

The EW receiver may also be required to "monitor" a radar that has been already detected. The monitor process checks to see if the radar is still on. A substantial reduction in the use of the EW receiver is realized if the radar's revisit time can be predicted using previously detected results. It is also possible to have an adaptive process that uses feedback of successful "re-detection" events. In this case, the EW receiver can be tuned to intercept revisit of the radar. Note, that if the EW receiver is used to monitor radars that have been detected, some of the time needed for detection must be sacrificed. In this case, the SS must be modified or some dwells must be preempted to allow the receiver to be used.

Guidelines for Scan Schedule

Detection is the most demanding in terms of receiver use. If detection is too focused, it may miss events.

The reasons to change detection SS are: load share with other platforms, demand of monitoring or

Measuring tasks, need to focus on associated emitters, or new sets of radars.

Detection SS are fixed prior to the mission and either switched or computed during the mission. The switching rate is on the time scale of seconds or minutes. Detection of previously undetected radars does not involve highly dynamic (updated or controlled on a pulse by pulse) update process. A complication occurs when the receiver is used for monitoring or measuring known radars.

If there is no prior data about threat radars, the detection SS is a simple process that dwells on the radar bands in an unbiased fashion.

Definition of Terms

A scan scheduler produces a sequence of tuning commands for radar warning receivers (RWR) also called an EW receiver. These tuning commands are referred to as control descriptor words (CDWs). As a minimum, a CDW selects the frequency band and specifies a dwell time. The CDW can also set other receiver parameters such as gain, bandwidth, mode, and other control parameters that need to be set on a dwell by dwell basis. However, for the initial SS work it is assumed that the CDW only contains a band number, dwell time and threshold. If a CDW is sent to the receiver, the process that generated CDWs must wait until the dwell time is up before sending the next CDW. Typical dwell times are as small as 10 microseconds and the CDW generation rate is at the 0.1 MHz rate.

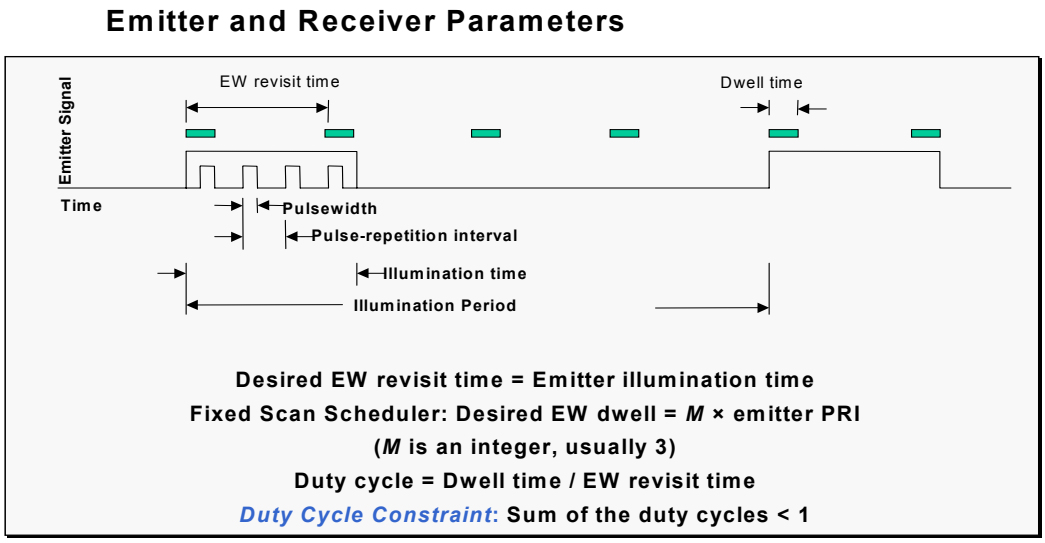
Pulses that are detectable by the RWR are called EDWs. The RWR generates a PDW for each detected EDW. A high-speed pulse processor sorts the PDW and identifies radars. As a minimum the PDW contains a time of arrival (TOA), pulse width (PW), amplitude (A) and frequency (f). The PDW may also contain other information such as direction of arrival (DOA).

Simple radars emit pulses at a fixed time interval called the pulse repetition interval (PRI). The radar also has an illumination time and illumination period. The RWR processing hardware must extract the PRI from the pulse TOA values. The RWR must detect at least 2 or 3 PDWs so that the PRI can be obtained. For more complex PRI patterns, the RWR needs to get several PDWs to determine the pattern.

To detect the radar, the RWR must dwell at the proper band for a time that is long enough to catch two pulses. Furthermore, the RWR must revisit the band often enough. If the RWR dwells for at least 2 radar PRI and revisits more frequently than the radar's illumination time, then detection of the radar is certain. Note that the radar's illumination time is a function of the radar type as well as the sensitivity of the RWR.

Figure 3 shows a timing diagram of a RWR and radar. Note that the RWR's revisit period is less than the radar's illumination period. This is one condition that is required for 100% detection probability of the radar. The other condition is that the RWR dwell must be at least one radar PRI. In figure 3 this is not the case. For classification the RWR dwell must be long enough to overlap two radar pulses. If the radar has a stagger pattern, the RWR may need to dwell for several radar pulses.

FIGURE 3 Emitter and Receiver Parameters



Often, a given band may have several radars operating at a time. In this case, the RWR processor must sort the pulse trains. This process is called de-interleaving. Note that many radars have complex PRI patterns. The pulse train sorting used PW, DOA, FMOP, Amplitude, frequency as well as TOA and stagger patterns to conduct the sorting process.

The RWR processor has an emitter data table (EDT) as well as an active emitter table (AET). The EDT contains all the emitter types that could be expected during the mission. For each emitter type, the EDT has information such as PRI (or stagger pattern for non-constant PRI), frequency (or frequency agility pattern), FMOP and other emitter data (priority for example). The initial SS is based on the EDT. The RWR processor matches detected data to the EDT to classify the radar. The AET contains a list of detected radars that have been detected.

Real Time Constraints

In ANTs, the objective is time critical resource management. Following are some guidelines:

(1) CDWs are generated at up to 0.1 MHz rates. They are typically not generated in response to PDWs on a PDW by PDW basis. Instead, simple generators generate them. However, if a radar has been detected, the time phase of a CDW might be synchronized the threat radar's pulse train (for advanced RWRs). If the SS is adaptive, the expected update rate is determined by the frequency of the

following types of events: radar is detected, missile is launched, radar mode changed radar stops operation, or aircraft maneuvers. These events will occur at a rate of 0.1 to 1.0 per second.

(2) PDWs occur at higher rates than CDW. Each emitter (in the band) will produce a few (at least 2 or 3) pulses in the dwell. If multiple emitters are in the same band (say 10), the PDW rate might be 20 to 30 times the CDW rate. The RWR processor (which is usually special purpose hardware) processes the PDWs to update the AET. The AET and EDT are used to update the SS. In some cases, the CDW times might be locked to the expected radar pulse TOAs. This is called pulse train tracking. It is similar to a phase locked loop and is used in some jamming functions.

The scan schedule approach must meet the following requirements:

The Scan Schedule shall be able to detect emitters. The detection process shall make use of the following emitter parameters: revisit time, band, and detect/classify time. Up to 50 different types of emitters may be possible. A maximum of 100 emitters may be operating. The maximum number of instances for each type is unbounded but is expected to be 10 or less. These limits are typical of the environment but should only be used to select the array bounds or memory size of the arrays or tables. Thus the limit of 10 instances for each type should not be used to stop the detection process for a specific type after 10 emitters of that type have been detected.

The scan schedule shall allow special DF measurements to "preempt" the normal detection process. If there are two platforms, the platforms must each make a DF measurement within a specified time window. An independent process will specify this window. The SS must generate a CDW to cause the DF measurement to take place.

The SS shall monitor emitters that have been detected and determine when the emitter changes its mode or stops operation. The monitoring process shall provide windows of the expected radar revisit time to a high level process that will request DF measurements.

The SS shall be compatible with the BAE SS demo hardware.

Detection performance is Pi (probability of detection on illumination I, I=1,2,….)

Monitor performance is T = Time of state change – time of detecting state change

DF measurement shall be made within the requested time window (the % of successful requests will be tracked)

The SS shall (if it is unable to cover all emitters for detection on the first illumination) allow priorities to be assigned. It shall provide (based on these the priorities) an estimate of the expected detection performance (assuming that only detection is being done – no monitoring or DF). Note an emitter will be detected if the CDW dwell overlaps the radar (emitter) dwell for a time of at least DD. The emitter will be detected on the first emitter dwell if the receiver revisit time is less than the radar's dwell time. The radar's dwell time is a function of the beam width, receivers threshold and the signal level (range related)

The SS shall take advantage of multiple platforms. Initially 2 platforms will be used. It shall be expandable to three or more platforms in the future.

The multi platform SS shall gracefully degrade to stand alone single platform operation in the event of loss of a platforms or loss of a comm links. Each single platform shall handle all the emitters handled by the multi platform (with possible degraded performance).

The multi platform solution must have a clearly defined API so that if any platforms are selected to run in "stand alone mode" emitters can be assigned and prioritized for that platform. The API shall allow multiple approaches to the "high level" multi platform control. The API must also permit DF measurements to be requested.

The SS software shall be able to drive the demo FPGA by loading a table of CDWs. CDWs may be fed to the RX model (from the FPGA table at a rate of up to 10**5/second. The FPGA has 2 CDW tables and can double buffer when a new table is to be used. There can be no delays after one table is "processed" and a new table is started. Potential delays might be waiting for the Dell PC to load a new table. Note that it is permitted to recycle (start over) the current table without loading in a new table. The API to the FPGA is defined by the CDW format and the FPGA table length (100 CDWs)

EW Simulator Data Generation and Parsing to the Firebird

The JSF simulator generates the emitter environment data used in the ANTs Scan Schedule Demonstration. The input to the simulator is provided from declassified emitter tables. Emitter Tables include emitter location and operational characteristics. The emitter table, along with an ownship flight path through the emitter lay down, produces a summary report consisting of the following eight (16 bit) parameters (except Time has 32 bits):

Time	0 to10 Sec. in 1-10mSec snapshots
Emitter ID	0 to 50 later up to 100
Frequency	2 to18 GHz (32 500MHz Freq. Bands)
Amplitude	0 to – 200 dB at the ownship surface
Pulse Width (PW)	0 to 1 mSec.
Pulse Repetition Interval	0 to 6 Sec.
Dwell Duration to Detect	typical 3 x PRI or a stagger period
Frequency Band.	1-15

A report is generated for each emitter ID (currently 1-50) at 10 millisecond intervals (0, 9,19,29,…,9990, 10000) for the duration of the 10 Sec. mission.

The summary reports are accumulated in an ASCII file for further processing. Static values such as ID, Frequency, PRI, Duration Dwell & Frequency are stored in a template configuration file while dynamic values such as Amplitude are masked onto the template config file and updated in 10 millisecond Snapshots.

The Amplitude is considered valid if its value is above a predetermined threshold. Otherwise it is reset to zero. Only emitters with non-zero amplitude values are processed in the FPGA.

As part of the preprocessing task, the ASCII summary report file is converted to a binary file. The binary file is read in by the ANTs simulator process and parsed down into 10 millisecond EDW snapshots. NEU/SRI provide 10mSec CDW snapshots populated with Time, Dwell Duration and Freq. Band.

The simulator broadcasts the snapshot data over a TCP/IP socket via Datagrams. These are synchronized to the FPGA using a HW generated 10 millisecond interrupt. Once the first memory bank has been filled the Simulator sends the HW a PROCESS_GO enable. Once the 10 second mission is completed the SW sends the HW a PROCESS_STOP enable (which is the inverse of the PROCESS_GO).

Several other registers are passed between the Simulator and FPGA such as for retrieving results the HW will register out the number of detects each represented by a 20Mbyte CDW including detects. This will allow the results read to be as deep as the number of detects.

Every EDW, CDW and PDW has a time stamp for cross correlation of results vs. simulation data. PDW results are in the same 20 byte word as the EDW format but now include the number of detects field.

PDW results are stored in the shared area, where the EDW simulation data and CDW data is also stored for NEU/SRI ANTs post processing analysis and future CDW generation using measured results for statistics and metrics.

Control Based Scan Scheduling-Overview

This section presents details and of the NEU control-theory based scan scheduler (CBSS). The specific task of the CBSS is to control a sensor that monitors sources of electronic radiation in the environment (such as radars). The sources emit radiation in different frequencies and with various illumination patterns and periodicities. For the sensor to detect an illumination by an emitter, the sensor must be tuned to a frequency band containing the frequency being emitted when the illumination occurs. The goal is to minimize the time spent attempting to observe each emitter, while also maximizing the probability of detecting an illumination by the emitter. The environment is dynamic since both emitters and sensors are moving.

The CBSS approach is formulated as a control problem by modeling the environment as a dynamic system and by specifying the required behavior in terms of a quality of service (QoS) metric. It consists of a two-loop feedback control architecture and a real-time scheduler to solve the problem of scheduling the receiver. The performance advantage of CBSS is that it can adapt to dynamic changes in the environment.

Control Based Scan Scheduling--Introduction

Modern avionics systems carry a number of sensors for collecting data about the environment. Normally the demands for sensing are much higher than the physical capabilities that the sensors can provide. As a consequence, the sensors must be used selectively, balancing the conflicting sensing requests from the point of view of the goal of the mission of the avionics system. The research domain that deals with controlling sensors is called Sensor Management. The goal of a sensor management system is to select the right sensor to perform the right service on the right at the right time.

In this section we describe the problem of controlling a sensor that monitors (detects) sources of electronic radiation in the environment (such as radars). The sources emit radiation in different frequencies and with various illumination patterns and periodicities. For a sensor to detect an illumination by an emitter, the sensor must be tuned to a frequency band that contains the frequency being emitted, and the sensor must be sensing at the time when the illumination occurs. Because of the large number of actual and potential emitters in an environment, it is important to minimize the time spent attempting to observe each emitter, while also maximizing the probability of detecting an illumination by an emitter. It is also important to keep frequency bands available for other uses, such as jamming. This results in additional pressure to reduce the time scheduled for the sensor to perform observations in each frequency band.

A generally agreed upon model for sensor management is to view it as a loop consisting of three major functions: {*generate options, prioritize options and schedule tasks*}. The first function generates request for sensor tasks, the second assigns priorities to the tasks, and the third then assigns the time of execution to each task. All three functions are addressed by the CBSS approach. Since in the subject domain the solution to this sensor management problem is termed as "scheduling", we use this term to mean inclusively "option generation, prioritization and scheduling".

The environment that must be addressed is highly dynamic. Emitters are rotating and the sensor is located on a moving platform. In the general case, emitters can move, and illumination patterns can vary over time. Furthermore, events can occur over a very large range of time scales. Individual radiation pulses may be as brief as a nanosecond, while illumination periodicities can be as long as several seconds. Additionally, since the scheduler is decoupled from the task generation and prioritization functions, the scheduler may introduce delays in scheduling the sensor.

To compensate for the dynamics of the environment and of the system, a control based resource manager (called a Dynamic Scan Scheduler (DSS)) is used. In order to map the problem onto the control architecture, we must also have a well defined quality of service (QoS) metric that specifies the required level of service. In addition to providing a metric for comparing the quality of various algorithms, the QoS is used as part of the control architecture. Various choices for the QoS are considered and discussed later.

The next step is to choose a suitable control architecture. Our methodology is based on a general architecture for self-controlling software which is specialized to this particular resource allocation problem. In this case, we chose a two-level feedback control architecture.

Basic Background of Emitters and Sensors

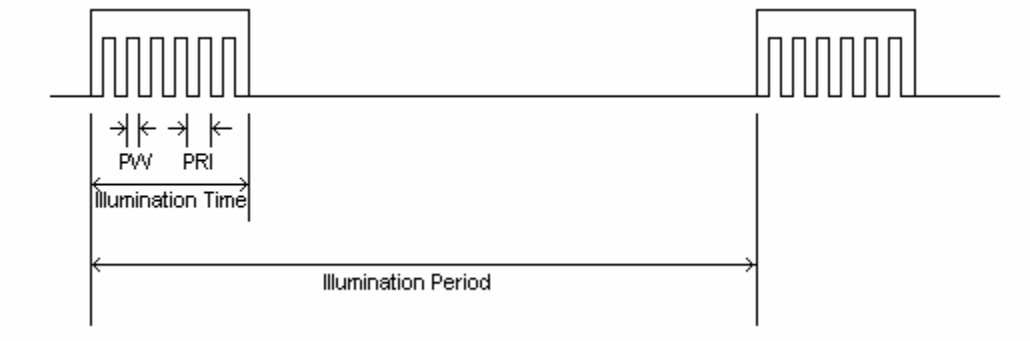
The sensors are receivers located on a moving platform such as an aircraft. When aircraft are operating in hostile territory, attempts will be made to detect and to track their movements. This is most often done by using ground or missile based radars. Such radars emit electromagnetic radiation that illuminates the aircraft. Radiation reflected from the surfaces of the aircraft is used by the radar for detection, identification and tracking purposes. We refer to these radars as emitters.

Because of the tactical importance of these radars, it is important for aircraft to detect that they are being illuminated. For this reason, aircraft are equipped with receivers that attempt to sense when the aircraft is being illuminated by an emitter. The aircraft receivers will be called sensors. A sensor can detect an emitter if the sensor is within the beam width from the central beam line of the emitter's antenna.

It is normally assumed that the kinds of emitter that might be encountered during a mission are known in advance. In particular, the system has basic data on the frequencies used by the emitters and the times between successive illuminations when the emitters are operating. It is not normally assumed that one will know when a given kind of emitter will begin operating or how long it will operate.

In Figure 4 we show a typical emitter illumination pattern. The radiation is produced in short pulses. These, in turn, form batches called illuminations. The pulses are shown here as square pulses, but the actual pulses are high frequency waveforms. The line over each illumination group is meant to represent the envelope of the illumination. It is not another signal.

FIGURE 4 Radar Scan Pattern



While the pulses are produced electronically, the illuminations are usually the result of the mechanical motion of the radar antenna. This has some important consequences. The time between pulses can be very short and can be very regular. On the other hand, the time between successive illuminations is typically much longer and need not be as regular.

The most important emitter parameters are the following: *fe* - the frequency of the emitter's signal (waveform); A_e - signal strength (power level) of the emitter; T_{pri} - Pulse Repetition Interval (PRI) of the emitter (time between successive pulses within a single illumination); T_{pw} - pulse width (PW) of the emitter; T_{eip} - the illumination period (time between successive illuminations, determined by the rotation period of the radar); B_e - beam width of the emitter (angle scope within which the emitter is radiating at a given time, determines the angle within which a target can be detected); T_{eit} - illumination time (duration of a single illumination); τ_e - the minimum time required for a detection of the emitter against background noise.

Sensor Control and Scheduling

A sensor is controlled by giving it a command called a Control Description Word (CDW). A CDW specifies when and for how long the sensor is to be tuned to a particular frequency band. Each CDW commands a single dwell of the sensor. Mathematically, a CDW is a triple:

$$\text{CDW} = (t_{ds}, \tau_d, \omega)$$

where t_{ds} is the time when the dwell begins and τ_d is the length of time that the receiver devotes to this dwell. A scan schedule (SS) is a sequence of CDWs, one per dwell:

$$\text{SS} = \{\text{CDW}_1, \text{CDW}_2, \ldots, \text{CDW}_n\}$$

A scan schedule is simply the sequence of commands given to a receiver. Within a scan schedule, one can define the notion of the *revisit time*, τ_{rv}, the time from the beginning of the last dwell until the beginning of the next dwell on a particular frequency band. From the beginning time of the last dwell and the revisit time, one can compute the start time of the next dwell.

The most basic constraint that any sensor must satisfy is called the capacity constraint. This constraint simply states in mathematical form the fact that a sensor can only be tuned to one frequency band at a time. For a scan schedule for which the number of frequency bands is a fixed number N and for which the dwell time and revisit time for each frequency band *i* is fixed, the capacity constraint is:

$$\sum_{i=1}^{N} \frac{\tau_d(i)}{\tau_{rv}(i)} \leq 1$$

Another way to view the capacity constraint is introduce the notion of the duty cycle. For each emitter, the fraction of the time allocated to this emitter is the emitter's duty cycle. It is the ratio of the dwell time to the revisit time for this emitter. The capacity constraint states that the total of all duty cycles can be no larger than 1.

While the capacity constraint is a necessary condition, it is not sufficient to ensure that a scan schedule exists. One limitation is that sensors require a small amount of time to switch from one frequency band to another. Another limitation that is more subtle is that the revisit times of different frequency bands may cause scheduling conflicts: two or more frequency bands may require the sensor at the same time. For this reason scheduling is an important part of the control of a sensor.

When an emitter has been detected, an attempt is made to identify the kind of emitter from a database of potential emitters. If this is successful, then the entry in the database is transferred to a table called the *Active Emitter Table* (AET). The AET is the primary output of the sensor, and it can also be used for scheduling the sensor.

If the emitter cannot be identified as a known kind of emitter, then the system creates a new AET entry as well as a new database entry. This usually requires that the emitter be detected more than once in order to obtain an estimate for the illumination period. Needless to say, such an unanticipated kind of emitter has important tactical and strategic consequences.

Introducing Dynamics to a Scheduler

One way to lower the number of dwells without missing (not detecting) illuminations is to synchronize the dwells with the illumination period of a given emitter. This is possible only after first detection of that emitter. However, in the dynamic environment we still are risking missing illumination due to dynamic effects. To compensate for the dynamics of the environment, a *dynamic scan schedule* (DSS) is used. Rather than being a fixed scheduling pattern, a dynamic scan schedule is an algorithm that dynamically modifies the scan schedule in response to detection events.

There are several ways that one can introduce dynamics into a scan schedule. One possibility is to use a precomputed FSS until first detection, and then recompute an FSS so that the newly detected emitter is only observed at the predicted illumination times as specified by the entry in the AET. After recomputation the CDWs for the emitter differ only in the start time t_{ds}, i.e., all the dwell times τ_d for the frequency band are the same. We will call this technique *informed dynamic scan scheduling*. The only difference between this technique and FSS is that detected emitters will have a much longer revisit time.

While informed dynamic scan schedule seems like it solves the problem of resource allocation in this case, it has some serious disadvantages. The main problem is that the illuminations of the aircraft will vary over time for a variety of reasons. The radar antenna, being a mechanical device, could speed up or slow down. However, a more significant effect is due to the motion of the aircraft. As the aircraft moves relative to the emitter, the apparent time between illuminations changes. For instance, consider an emitter with a 6 sec illumination period (i.e., the radar antenna rotates by one full cycle in 6 sec). Now suppose that the aircraft is 60,000 feet away from the radar and is moving in a circle around the radar at 1000 feet/sec, in a direction opposite to the rotation of the radar. In such a case while the radar rotates by 360 degrees, the aircraft moves by 6,000 feet, which is equivalent to about 6 degrees. Consequently, the apparent time between illuminations will be shorter by approximately 0.1 sec. This might seem like an insignificant shift, but since the sensor dwells on a particular target for the duration of just three-pulse repetition intervals, say $T_{pri} = 6 \cdot 10^{-3}$ sec, this shift is actually quite important. In fact, this dwell time is only about 6% of the 0.1 sec shift, which is more than sufficient to cause a detection failure. To address the problem of dynamic changes in the environment, we propose a feedback-control based DSS.

Mapping to a Control Architecture

We now show how the sensor scheduling problem can be mapped to a control based architecture. We first review some background in the control theory metaphor of software development. A specific mapping to a control theory architecture is then presented.

Control Theory Metaphor for Software Engineering

The approach is based on a control theory metaphor for software development. The reason for proposing such an approach was to address the problem of changes in software requirements. In the case of embedded software, changes in software requirements come from changes in the environment with which the software interacts. Fixed scan schedules do not perform well when the environment changes dynamically or the workload is heavy. This is due to the high duty cycle required for each potential emitter and the relatively low probability of detecting each emitter. The control theory paradigm seems natural for this kind of application.

The main idea behind the control theory metaphor is to treat the basic software functionality as a Plant, to treat changes in the environment as disturbances, and to use feedback from the environment to adjust system behavior. In order to compensate for disturbances, a Controller must be added. From the point of view of software requirements, the Controller is a redundancy that is added to the system. The controller is not part of the original software requirements. Other kinds of (redundant) modules need to be added to implement a control loop, as described below.

In the control approach a system consists of two basic elements: a Plant (the system being controlled) and a Controller. To map a problem to a control architecture, it is sometimes necessary to add a Quality-of-Service (QoS) module that assesses the Plant's output in terms of a Quality-of-Service measure. (Note that in the control literature, QoS is not usually regarded as a separate module; it is simply a part of the controller.)

Plant

The whole system (the plant) for this problem includes a receiver (sensor), a sensor management module and an environment (a number of potential radar emitters). The receiver is traveling through the environment on a platform. The platform is usually an airplane.

The sensor management module consists of three functions: *prioritize, generate options,* and *schedule*. In our mapping, the first two functions are included in the module called *CDW Generator*. In the following, we specify all these modules.

Emitters and Receiver

The state of the platform and its receiver at a time t is determined by the position of the platform and by the frequency band to which the receiver is tuned. The position of the platform is determined by its Cartesian coordinates $(x(t), y(t))$ over the terrain. The altitude of the platform is not significant for this problem domain. The trajectory of the platform can be arbitrary but it is fixed in a particular mission so it is modeled as the

input process of the dynamic system. The position as well as the velocity $\left(x^1(t), y^1(t)\right)$ of the platform can therefore be regarded as known.

The platform has a receiver that can be tuned to various frequency bands. It can only detect an illumination by an emitter if the receiver is tuned to a frequency band that includes the frequency of the emitter's signal. The state of the receiver can be represented by a finite discrete set of frequency bands, along with one point representing the state in which the receiver is not tuned to any band. The current frequency band is written ω, and the set of all such bands (including the ``off'' state) is written Ω. The state space for the platform is therefore the product of the (continuous) two-dimensional plane and the set of frequency bands:

In contrast with the platform, we will assume that the emitters are stationary while emitting and that they run independently. A state for a single emitter consists of the following three variables:

(1) The direction ϕ in which the emitter is illuminating its environment. The variable ϕ is an angle, and therefore its set of values is isomorphic to the unit circle or to the half-open interval $[0,2\pi]$.

(2) The distance r between the platform and the emitter.

(3) The angle θ of the platform with respect to the emitter.

The state space for emitter i will be denoted $E(i)$, and the state of the emitter at time t will be denoted $s(i,t) \in \mathrm{E}(i)$. The variables r and θ represent the position of the emitter in polar coordinates. It is sometimes convenient to use Cartesian coordinates, in which case they will be written $\left(x_e(i,t), y_e(i,t)\right)$.

The evolution function of the system has both continuous and discrete components. The continuous component evolution is determined by the following stochastic differential equation for emitter *i*:

$$\frac{d}{dt}\phi(i,t) = 2\pi / T_{eip} + \omega_{\phi}$$
$$\frac{d}{dt}x_e(i,t) = x'(t) + \omega_{xe}(i)$$
$$\frac{d}{dt}y_e(i,t) = -y'(t) + \omega_{ye}(i)$$

In this system of equations, ω_{ϕ}, $\omega_{xe}(i)$ and $\omega_{ye}(i)$ are the noise affecting the respective derivative. These are assumed to be Gaussian white noise (with mean 0). It is also assumed that ω_{ϕ} is independent of the other two noise variables.

The discrete component of the evolution of the receiver is characterized by discontinuous ``jumps'' from one state to another. The receiver evolution is determined by the scan schedule. Each CDW in the scan schedule generates an event, which in turn, causes a discontinuous jump from one frequency band, ω_i, to another, ω_j, at the time of the event.

The last component of this dynamic system is the output function. This function determines what function of the variables of *Q* are observed as the system evolves. As is typical of such systems, the objective is to infer the values of the state variables from what can be observed. The output function computes *detections*. Whenever the receiver satisfies the requirements for a detection, the function gives the data associated with the detection.

For each emitter there is an upper limit r_m on the distance at which it can be detected. When the platform is beyond the range of the emitter, then no detection is possible at any angle. The same would be the case if the emitter were not emitting either because it was not turned on or because the emitter was not deployed. For the purpose of this problem, it is not possible to distinguish an out of range emitter from one that is off.

The following are the requirements for detecting emitter *i*: the emitter must be operating and within range, the emitter must be illuminating the receiver, and the emitter frequency must be in the current receiver frequency band. The requirements above must hold for a period of time at least equal to τ_e.

The receiver actually measures the amplitude of the illumination continuously during the dwell time, and it determines that an illumination has occurred by comparing the

measured signal with a noise threshold. A more elaborate model would, of course, consider the measurement noise in addition to the process noise.

When the receiver dwells on a specific emitter and does not detect that emitter, the output function returns $t_l(i)$, i.e., the *time of last look* at that emitter. The output function produces the set of detection times and look times for all emitters that the receiver looked as directed by the scheduler.

For simplicity, we will say that the value of the function is either t_d or t_l. Since the value of this function will not change between dwells, the set of detection times will be denoted as T_d and the set of non-detections will be denoted as T_l.

Note that this definition assumes that when more than one emitter is transmitting on the same frequency band, the receiver can distinguish them from each other. In other words, it can determine which of the potential emitters in a band is actually illuminating the receiver. Because emitters differ from one another in the detailed structure of the illumination (i.e., the pulse pattern), this is not an unreasonable assumption, although it might not always be correct. If such discrimination is not possible, then the output function must be redefined so that it only computes the frequency band that is being detected, not the specific set of emitters.

Since a detection can only occur during a dwell, one simplification of the output function is for the output function to produce a sequence of data structures, one for each CDW. This is, in fact, what many receivers produce. The output is a data structure called a {\em pulse descriptor word} (PDW) which contains detailed information about the illumination(s) that were detected during the period of time covered by the CDW.

CDW Generator

Another part of the plant is CDW Generator. Two parts of the CDW are fixed for each emitter: τ_d and ω. The time of the start of the next dwell, $t_{ds}(t+\Delta t)$, is re-computed at the time of the assumed beginning of the current dwell, $t_{ds}(t)$. Since the CDW Generator shifts starts of dwells, this module also introduces additional dynamics to the system.

If the emitter illumination was perfectly periodic, if there were no delays, if both the platform and the emitter were stationary, and if we knew the distance between the emitter and the platform then we would be able to predict exactly the time of the next dwell as the time of the current dwell, $t_{ds}(t)$ (assuming the receiver dwells on each illumination),

plus the revisit time for that receiver, τ_{rv}. In order to compensate for delays, we need to adjust τ_{rv}. We use the following model to achieve this goal.

$$\tau_{rv}(i,t+\Delta t)=\left\{\begin{array}{c}\tau_{rv}(i,t),\quad if\ t<t_{ds}(i,t)+\tau_{rv}(i,t)\\ \delta'(i,t)^2\cdot T_{eip}(i),\ otherwise\end{array}\right\}$$

$$t_{ds}(i,t+\Delta t)=\left\{\begin{array}{c}t_{ds}(i,t),\quad if\ t<t_{ds}(i,t)+\tau_{rv}(i,t)\\ t+\Delta t,\quad otherwise\end{array}\right\}$$

$$\delta'(i,t)=\left\{\begin{array}{c}1,\quad if\ \delta(i,t)>1\\ \Delta(i),\quad if\ \delta(i,t)<\Delta(i)\\ \delta(i,t),\quad otherwise\end{array}\right\}$$

$$\Delta(i)=\left(T_{eit}(i)/T_{eip}(i)\right)^{0.5}$$

In these equations δ is the input from the Emitter-Level Controller. The value of $\Delta(i)$ determines the smallest revisit time. If the system always used the revisit time determined by $\Delta(i)$, it would be equivalent to a FSS. The time increment Δt used in our simulations was 10μ sec . It is the smallest ``dwell grain'' that makes sense for a particular system.

Scheduler

The CDWs generated in CDW Generator are fed into a scheduler's queue. However, since the CDWs are generated by a number of independent generators (one per emitter), it results into a too high demand for the receiver. The limit on the capacity of the receiver is defined by the instant workload; it is violated if the value of the workload is higher than one.

We use a priority-based scheduler, where priorities are computed dynamically. The scheduler is invoked at *schedule time*, t_s. The schedule time is defined as the completion time of the current dwell, i.e., $t_{ds}(t-1)+\tau_d(t-1)$ or the beginning of the next dwell in the queue.

A scheduling decision $\omega(t_s)$ is made based upon the current value of the *QoS,* where a specific frequency band j is chosen if the *QoS* for it is the largest.

$$\omega(t_s) = j,$$

$$QoS(j, t_s) = \max_i QoS(i, t_s)$$

The Quality of Service Metric

The *QoS* is a metric that determines the level of quality exhibited by the system at any point in time. It can be used to specify the required level of service of the system and to compare the performance of different algorithms. As noted above, the purpose of the system is to infer the state from the limited information in the output function. The sensor system is continually computing its current best estimate of the state variables. Obviously, it cannot know the state variables exactly, and not all state variables are equally important. In many situations, the most important fact about an emitter is that it is emitting within range of the platform, while in other cases this fact is less important than the location of the emitter relative to the platform. The first situation is known as the "detection problem" while the second one is usually called the "tracking problem".

Because the problem to be solved is the estimation of the state variables, the *QoS* function should measure the accuracy of the estimates, such that the more important state variables are given more weight in the metric than the less important ones. Of course, the actual accuracy is not known, but it is possible to estimate it.

Since the dynamic system for each emitter is a linear stochastic dynamic system, the overall measure of accuracy is the (co)variance. The dynamic system is 3-dimensional so the covariance matrix is a positive definite matrix of order 3. For emitter *i,* this matrix is block diagonal with two blocks as follows:

(1) Position uncertainty

The variance of the position estimate is primarily due to the motion of the platform, since the emitters have been assumed to be stationary. The position noise variance will necessarily vary with time since it depends on the direction of motion of the platform. The motion of the platform is a known (input) process, so its noise variance is known. We will assume that the uncertainty in the *x* and *y* directions are the same and are uncorrelated, i.e., that the covariance of the position uncertainty has the form $\sigma_p(i,t)I$, where *I* is the unit matrix, and $\sigma_p(i,t)$ is the common variance in both the *x* and *y* directions. In addition, we assume that the position noise variance does not depend on time.

(2) Beam direction uncertainty

The rotation of the emitter is a mechanical system so it is subject to mechanical noise. We assumed that this noise is independent of the other emitters and of the position noise. In addition, we assume that it is constant for each emitter. We write $\sigma_b(i,t)$ for the estimated variance of the beam direction of emitter *i* at time *t*.

By the assumptions above, both $\sigma_p(i,t)$ and $\sigma_b(i,t)$ increase linearly with time when emitter *i* is not being observed. The increase is due to the respective noise variables, and the derivatives of the variances with respect to time are the respective noise variances.

The *QoS* metric for each emitter is defined by linearly combining the position and beam direction variances: $QoS(i,t) = a_i\sigma_p(i,t) + b_i\sigma_b(i,t)$. The constants a_i and b_i determine the importance (priority) of emitter *i* relative to other emitters. These constants also determine whether the emphasis for emitter *i* is on detecting the emitter or on locating the emitter. We call $QoS(i,t)$ the *emitter-level QoS* metric.

When the receiver is tuned to a frequency band containing the frequency of emitter *i*, then the emitter is being measured. The measurement is extended over a period of time (the dwell time), and the output of the receiver is a continuous output over this period. The measurement is therefore a continuous-time stochastic dynamic system just like the platform/emitter dynamic system. Unlike the dynamic system describing the platform and emitters, which is a linear dynamic system, the measurement process is nonlinear. As a result of the nonlinearity, the gain due to the measurement varies with the beam direction as well as the distance from the emitter. For example, when the emitter is illuminating the platform, the gain will be much higher than when the emitter is not.

One can summarize these observations as follows:

(1) When emitter *i* is not being observed, its emitter-level *QoS* increases linearly with time.

(2) When emitter *i* is being observed, its emitter-level *QoS* decreases with time, but the decrease depends on the result of the measurement.

A similar model for sensor management was introduced investigated, but in that model the measurement process was assumed to be linear. As a result, the reduction in the variance was a linear function of time during a measurement.

We assume that the receiver measurement is processed by the receiver to produce a discrete two-valued response rather than a continuous amplitude response. This

simplifies the effect of a measurement on $QoS(i,t)$ to just two cases: a large gain when the receiver detects an illumination and a small gain when the receiver does not detect an illumination.

We now give a precise formula for the emitter-level *QoS* metric based on the discussion above:

$$QoS(i,t) = \frac{P(i)}{T_{eip}(i)} \cdot (t - t_d(i)) + d(i,t_l)$$

$$d(i,t_l) = \begin{Bmatrix} P(i) & : & t_l = StartTime \\ 0 & : & t_l \in T_d(i) \setminus StartTime \\ d(i,t_l^-) - P(i) \cdot \frac{T_{eit}(i)}{T_{eip}(i)} & : & t_l \in T_l(i) \setminus T_d(i) \end{Bmatrix}$$

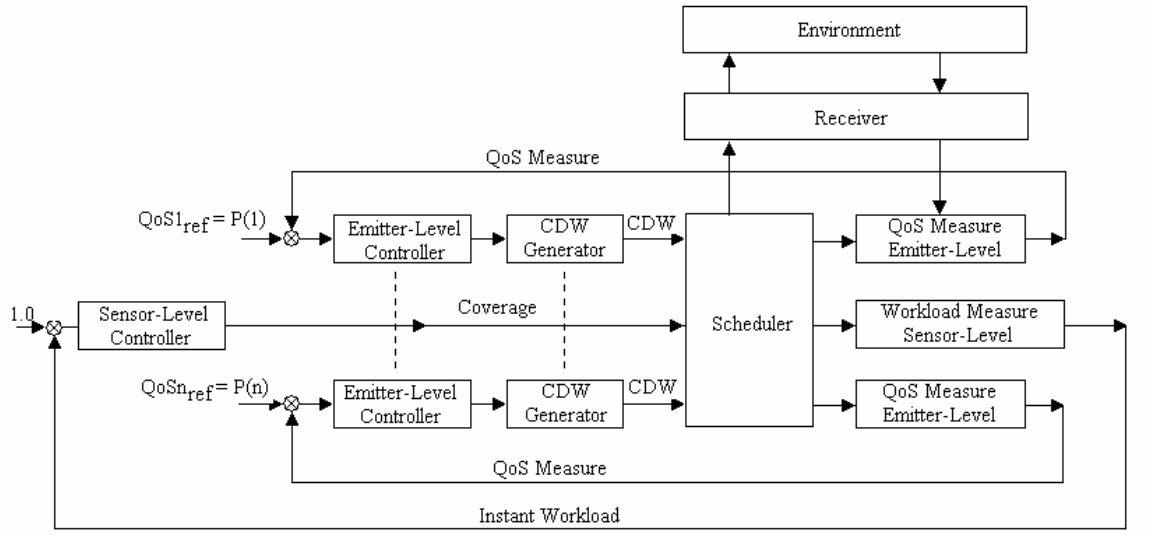
In this formula, $t_d(i)$ denotes the time of last detection of emitter *i*, and $P(i)$ denotes the weight associated with this emitter. $P(i)$ represents the importance of the emitter to the EW mission; it may also be thought of as priority of the emitter. In addition, $T_d(i)$ denotes the set of detection times for this emitter at the start time, and $T_l(i)$ denotes the set of observation times for this emitter. We consider the observation time to be the time at the end of the dwell, so $T_d(i)$ is a subset of $T_l(i)$. Note that the function $d(i,t_l)$ is updated only at observation times.

The emitter-level *QoS* metrics can be combined to form the overall *QoS* metric called the sensor-level *QoS*. The combination is usually either the maximum (or minimum) of the individual *QoS* metrics, or the (weighted) average of the individual *QoS* metrics.

Controllers

As shown in Figure 5, the system includes two controllers - an Emitter-Level Controller and a Sensor-Level Controller.

FIGURE 5 Control Based Scan Schedule



Emitter-Level Controller

There is one Emitter-Level Controller for each CDW Generator, i.e., one per emitter. The goal of this controller is to compensate for the delays due to both platform movement and scheduling. In the system described in this paper we simply used a proportional controller, with the proportional parameter denoted as $K(i)$. The control output $\delta(i,t)$ generated by this controller is a parameter in the CDW Generator module. The reference input for the Emitter-Level Controller should be set to an expected level of uncertainty. Since we have different priorities for particular emitters, we set the reference input to the value of priority of a given emitter, $P(i) \in [0,1]$.

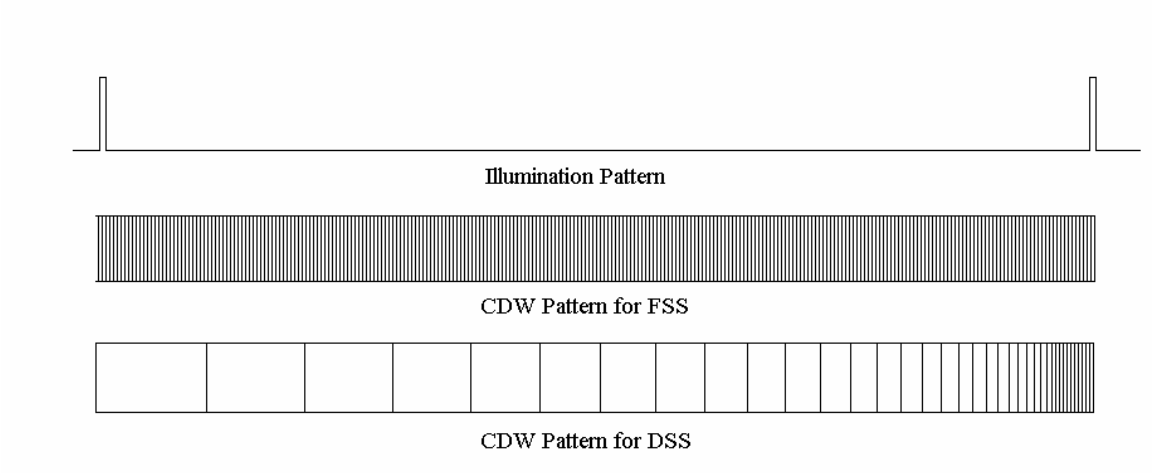
Sensor-Level Controller

The goal of the Sensor-Level Controller is to compensate for changes in the demand for the workload (overload) of the receiver. The *QoS* for the Sensor-Level Controller is the overall instant workload. The reference input for the Sensor-Level Controller is set to 1. It is the maximal capacity for scheduling. We used a PID controller for sensor-level control. The proportional, integral and differential parameters for the PID controller are denoted as K_p, K_i and K_d, respectively. The output of this controller is denoted as $c(t) \in [0,1]$.

Emitter-Level Controller: Design Issues

In this section we analyze the behavior of a scheduling system under various approaches to scheduling. First we discuss the FSS and then compare its behavior with a DSS. This analysis is intended to provide guidelines for designing controllers, i.e., selecting the control parameters K_p, K_i and K_d. First we focus on the FSS. The FSS dwells follow a fixed dwell pattern. The pattern is designed in such a way that the dwells are very frequent, only one emitter illumination time apart. An example of a FSS dwell pattern is shown in Figure (top). We can see from Figure 6 that only one or two of the 80 CDWs result in a detection. All other CDWs are useless for detection.

FIGURE 6 Timing Diagram



A pattern for a DSS is shown in the bottom part of Figure 6. This pattern is described by the model of the CDW Generator and by the Emitter-Level Controller. This pattern shows dwells generated by the CDW Generator after the first detection of an illumination. The time span is equal to the time between two consecutive illuminations.

The question is that whether such a dynamic CDW pattern will detect every illumination in spite of phase shifts due to the motion of the platform? It is possible for the shift in apparent time between illuminations to be 0.1 sec, which is about 2% of the receiver illumination period. This is equal to three illumination times and thus it is not negligible. In order to detect an illumination, the revisit time within the vicinity of the illumination must be less than or equal to the value of illumination time minus dwell, which in this

example is about 0.5% of the illumination period. Consequently, when $t_{ds}(i,t) - t_d(i) > 0.98 \cdot T_{eip}(i)$ the condition that guarantees detection is:

$\tau_{rv}(i) < 0.005 \cdot T_{eip}(i)$

The dwell pattern depends on the dynamics of the plant and on the controller's parameter $K(i)$. Table 1 shows many CDWs are needed for one illumination period depending on the value of $K(i)$ and on the dynamics of the system (amount of shift in the illumination pattern).

$K(i)$	$\frac{t_{ds}(i,t) - t_d(i)}{T_{eip}(i)}$	Number of CDWs
0.08	0.1191	193
0.4	0.8268	63
0.6	0.8851	42
0.8	0.9125	31
0.98	0.9604	9

Table 1: Number of CDWs for Different Controllers and Dynamics

If we knew the distance to a given emitter, provided that we know the illumination period of the emitter and the speed of the platform motion, we should be able to pick an appropriate $K(i)$, i.e., such that guarantees detection while not overloading the receiver with too many requests for dwells (number of CDWs). However, if this kind of knowledge is not available, we need to select a more conservative approach. For a low dynamics system we can choose a controller with a large $K(i)$ resulting in fewer CDWs. For a system with higher dynamics, we can choose a controller with a small $K(i)$, which can generate more CDWs but will give a higher probability of detection.

Sensor-Level Controller: Design Issues

Assuming duty cycles for three types of emitter are 0.108, 0.054 and 0.0015 *sec*, we can compute workload for a FSS. For instance, with 50 emitters of the first type, 50 emitters of the second, and 100 emitters of the third, the workload is 8.25. We call the load computed for an FSS the *static load*. The FSS scheduler is not capable of handling such a workload. If the dynamics of the system is like shown in the last row of Table 1, the DSS would need 9 CDWs per emitter, while the FSS would need 180. Consequently, the DSS should be able to handle a workload that is ten times higher than the upper workload for a FSS, which is 1. Therefore, the DSS should be able to handle the load of 8.95, which is lower than the upper limit of 10. While this would be true for a fixed pattern, the DSS changes the dwell pattern, and thus *instant load* at any specific time may be higher than the estimated static load. It is the role of the Sensor-Level Controller to adjust the

workload so that the instant workload does not go higher than 1. The discrete representation of the PID is given by the following equation.

$$C(Z) = K \cdot \frac{Z^2 + a_1 \cdot Z + a_s}{Z^2 - Z}$$

where

$$K = K_p \cdot (1 + K_i + K_d)$$

$$a_1 = -\frac{1 + 2K_d}{1 + K_i + K_d}$$

$$a_s = \frac{K_d}{1 + K_i + K_d}$$

The plant model can be approximated as:

$$P(Z) = \frac{1}{z - 1}$$

The close-loop discrete transfer function of the whole control system is:

$$H(Z) = \frac{K \cdot (Z^2 + a_1 \cdot Z + a_2)}{Z^3 + Z^2 \cdot (K - 2) + Z \cdot (1 + K \cdot a_1) + K \cdot a_2}$$

Based upon this model, we can design a system with poles and zeros $z1 = 0.91$, $pl = 0.885$, $p2 = 0.565$, i.e., such that the poles and zeros are located within the unit circle, which results in a stable system. Using Ruth method we find the PID parameters $K_i = 0.5$, $K_i = 0.1$ and $K_d = 0.0$.

SRI Rapid Scan Schedule Generation

SRI's approach is based on the ability to rapidly recompute the scan schedule in response to mission level events. Figure 7 is an overview of this approach. The approach is based on the ability to generate a scan schedule in a few seconds given a set of weights. When the weights change in response to mission level events, a new scan schedule can be generated.

The computation of the scan schedule must be performed under real-time constraints. The objective is to ensure that the maximal time for a module to compute a new schedule is of the order of a few seconds at most. The output of a scheduler module is a regular schedule that controls scan scheduling on the module's platform. This schedule is represented as a table of CDWs to be loaded in the FPGA-based receiver.

To develop this approach SRI conducted the following steps which are described in detail in the SRI final report:

(1) Develop a metrics to specify the expected scan schedule performance with a set of emitters.

(2) Determine the criteria that predict if a scan schedule that meets the performance criteria exists and can be found "soon enough" by the searching process.

(3) Develop a search heuristics that generate the scan schedule.

In the dynamic scan scheduling context, there must be guarantees that the schedule construction will succeed within the real-time constraints. Although the problem is NP-hard, there are many easy instances for which a solution can be found very rapidly. It was found that the total receiver utilization U is a good metric to predict the search time needed.

A key finding is that there is a phase transition around U=0.9, independent of the number of bands or emitters. Using a Monte Carlo approach to generate scan schedule requirements SRI found that almost all the instances with U more than 0.9 are infeasible and that almost all instances with U less than 0.8 are feasible.

The experiments were performed with a timeout of 90s CPU time. The number of instances that could not be solved within that time was determined as a function of U. We conjecture that all the instances that timed out are actually infeasible. This has been

confirmed experimentally for small values of n, by running the algorithm on the same instances without a timeout.

The experiments indicate that the cost of computing a regular schedule is dependent of U. Based on this the strategy for computing a good schedule within the delay D is as follows:

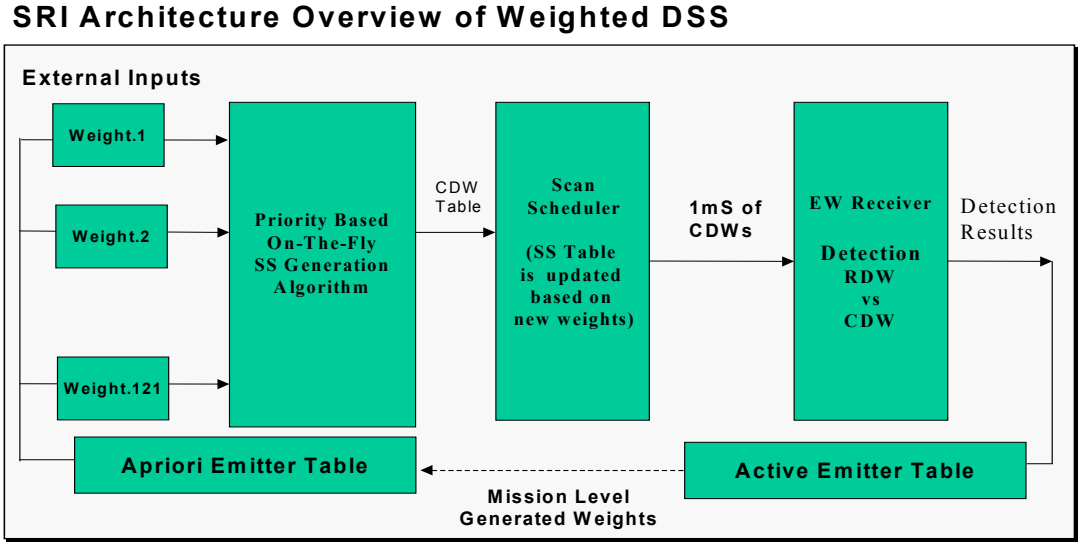
(1) Select an utilization threshold U for which P(U) is high.

(2) For this U, use the schedule-construction algorithm with a small timeout.

(3) If a schedule is found and D has not elapsed, attempt to improve the schedule by repeating the process with a higher value of U.

(4) If D has elapsed, repeat the process with a smaller value of U.

FIGURE 7 Rapid Scan Schedule Generation During the Mission



Extensions of SRI DSS to DDSS

The rapid scan schedule algorithm was employed in a multi-platform configuration by SRI. Figures 8 and 9 show how the SRI DSS approach can be extended to a distributed dynamic scan schedule (DDSS).

FIGURE 8 Approach to Distributed Scan Scheduling

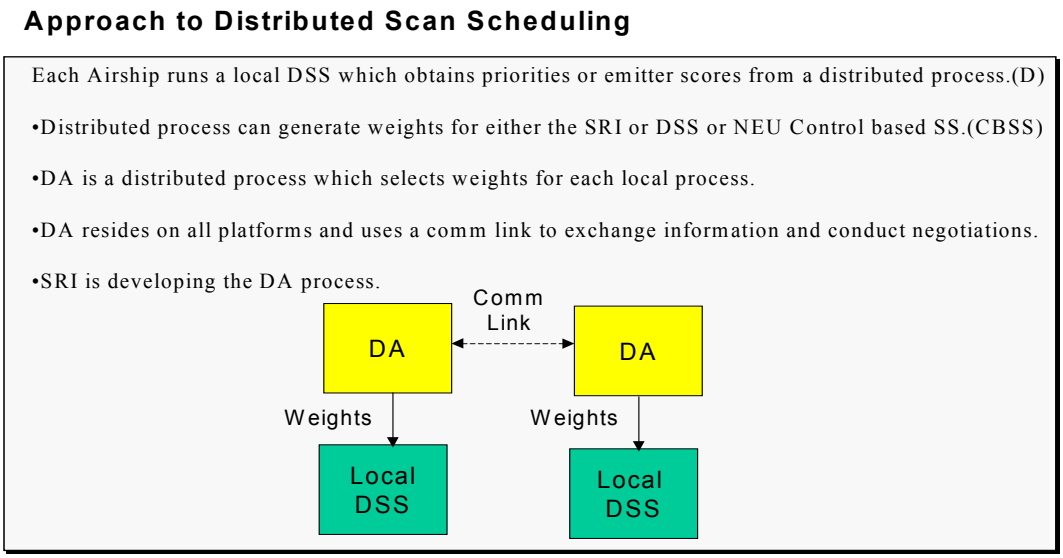
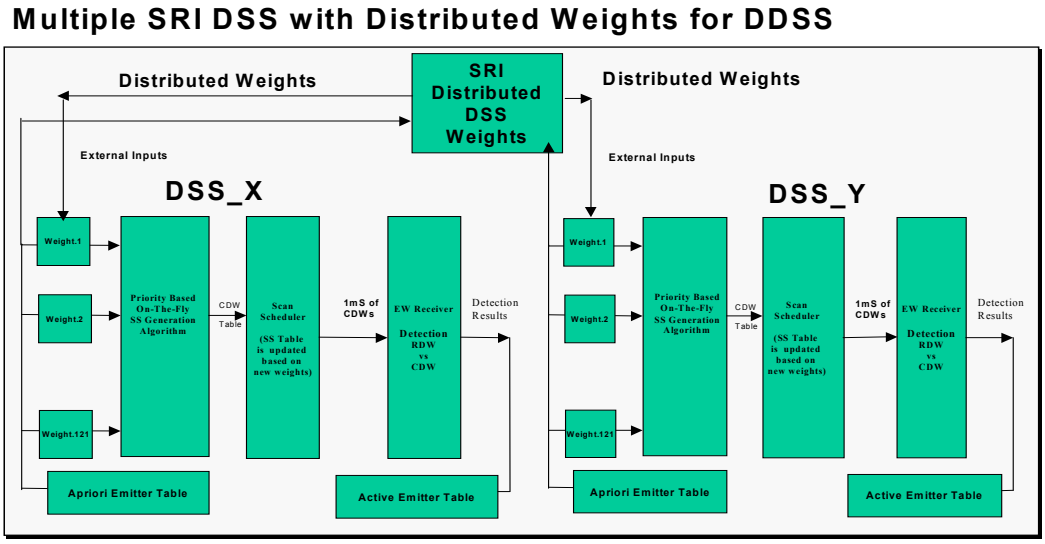


FIGURE 9 Multiple SRI DSS with Distributed Weights for DDSS



Test Results—Overview

Both the NEU and SRI approaches were tested on the BAE test bed. Both approaches appeared to have a rapid (or sharp) cutoff as the required ECM receiver load approached 100%. This was due to the fact that both approaches made the attempt to cover all threat radars instead of dropping some of the lower priority radars.

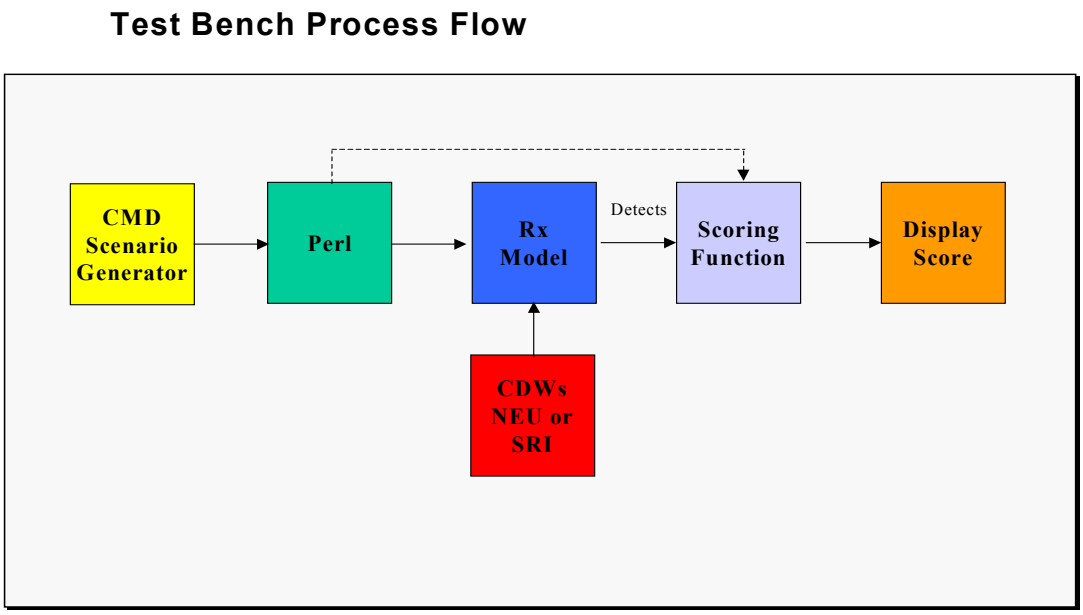
The SRI approach did improve the detection probability for cases that are near saturation (i.e. require 70 to 90% receiver use). However as the load increased beyond 90%, the SRI approach was no not able to converge to a solution. This problem can be fixed by allowing solutions that do not always cover the lowest priority radars with 100% coverage.

The NEU DSS approach did not offer significant detection improvement over the FSS. The NEU approach did show significant reduction in receiver use needed to "track" an already detected emitter. However, existing EW receivers use conventional approaches (phase locked loops) to do scan tracking of radars. Since the NEU approach requires feedback from detected events, it is not expected that it can improve detection. One can only get improvement in detection by making changes to the scan pattern that are not based on detection events. One possibility for improvement is to use the SRI approach to produce the initial SS each time a mission priority changes.

Test Bed

Figure 10 shows an overview of the SS test bed. The CMD scenario generator was obtained from the BAE countermeasures division (CMD). A perl script is used to convert the CMD output to a format that is used by the RX model. The RX model simulates the EW receiver and is controlled by the scan schedule process (red block). A scoring function is used to evaluate the scan schedule process. Finally, the results are displayed.

FIGURE 10 Test Bench Process Flow



The test scenario had 121 emitters. A sample of these emitters is shown in figure 11. Figure 11 also shows the emitter parameters. The scan schedule generation process must use this emitter information to generate the scan schedule.

FIGURE 11 121 Emitter Table

121 Emitter Table

	0-18GHz	nS	nS	dB			1-32 Freq	Sec.)
ID	FREQ/1K	PRI	PW	AMP	Startx	Starty	Bands	Scan Perio
1	8.037	3210	161	104	7362	-4891	13	11
2	7.54	3503	125	106	7868	-5042	12	12
3	4.721	331767	532	115	8581	-4062	6	3
4	5.188	362232	639	114	8508	-5498	7	3
5	5.291	347330	561	115	9242	-4491	7	3
6	9.008	644283	445	119	9332	-5428	15	19
7	8.708	380564	374	111	8637	-4830	14	16
8	9.146	363893	282	111	9182	-5765	15	19
9	8.836	412016	303	119	9377	-4962	14	17
10	9.011	589430	321	118	8210	-4657	15	17
11	8.816	620782	391	116	8983	-5622	14	17
12	9.558	462043	347	120	8026	-5565	16	18
13	9.361	527256	390	112	8290	-5197	15	16
14	8.787	581882	393	120	8377	-6041	14	20
15	8.785	514621	487	114	8309	-5679	14	14
16	9.096	627812	470	119	7900	-3958	15	20
17	8.998	360815	394	119	7394	-5868	14	15
18	16.535	232747	179	116	9556	-5543	30	10
19	16.503	160264	158	110	8285	-6312	30	8
20	16.813	199610	191	119	8953	-4593	30	8
21	16.588	210167	258	115	8821	-5659	30	12
22	17.132	147461	183	119	9481	-5530	31	11
23	16.5	187937	227	114	9064	-6250	30	11
24	1.935	128584	2707	87	9155	-5090	1	1
25	2.407	112169	3572	95	8466	-4199	1	2
26	1.717	106770	3672	90	8452	-4150	1	1
27	2.399	133909	2839	91	9147	-5398	1	1
28	1.775	160174	2685	88	8046	-4312	1	2
29	2.245	157483	2252	94	9290	-4146	1	1
30	2.635	139872	3329	95	7474	-5437	2	2
31	2.541	159721	2338	93	8129	-4897	2	2
32	1.682	170076	2826	87	7804	-3974	1	1
33	1.971	99811	2678	95	7842	-5115	1	2
34	1.784	166181	3882	88	7410	-4382	1	2
35	2.018	149639	2266	96	7295	-5011	1	2
36	2.377	115234	2327	89	8034	-4095	1	2
37	1.675	138686	2600	90	7343	-4520	1	1
38	2.058	114493	3814	87	6884	-4917	1	1
39	2.022	167296	3169	89	7411	-3860	1	1
40	2.068	99922	3902	91	7860	-4734	1	2

Emitter Table Contents:

Emitter ID (1-121)
Frequency (0-18GHz)
PRI (nS)
Pulse Width (nS)
Amplitude (dB)
StartX
StartY
Freq. Band (1-32)
Illumination Period (Sec.)

Note:
Illumination Angles 3& 10 degrees
Dwell Duration = 3 x PRI

Scoring Methods

The EW receiver must detect the radar as soon as possible. The objective is in the first illumination. Since the receiver is limited, the threat radars must be assigned a priority. The lethal radars have the highest priority. Often the set of lethal radars change as the mission unfolds. The details of the change is not known at the start of the mission. The dynamic scan schedule (DSS) must adapt to this change by updating the scan schedule.

The scoring method must give more credit for detection of critical emitters. Also detection must occur in a timely fashion. To accomplish a metric based on the following was used:

Maximum emitter score is obtained if detection occurs on the first illumination.

Detection on the second or third illumination results in lower emitter score.

Failure to detect a lethal emitter results in a negative emitter score.

The set of lethal emitters is not known in advance and may change during the mission.

The detection scores for specific emitter types are provided to the DSS during the mission.

Emitter scores are accumulated during the mission to form the overall score.

Figure 12 shows the details of the scoring method used to evaluate the DSS. The test consisted of running a mission that used a FSS to form a benchmark. Then the mission was repeated using either the SRI or NEU DSS. Scores for all three cases were generated.

In figure 12 note that some emitters (5 in this case) are considered lethal. Detection of these lethal emitters results in a 2000 point score increment. However, failure to detect after 3 illuminations results in a 10000 point loss.

If the scores are known in advance (before the start of the mission) there would be no need for a DSS for detection. However, the scores are not known in advance because the details that define which emitters are lethal change as the mission takes place. When emitters are detected, other associated emitters are expected to be present. Also, as more information is obtained, it is possible to deduce the intent of the enemy weapon system. This creates expectations that require a new scan schedule.

Finally, if multiple platforms are able to exchange data, the scan schedule may be modified in response to the new data. Multiple platforms can also share the scan load as well. In this case a DSS is essential because mission plans may change or a platform may be lost. The DSS will modify the SS in response to events that occur in the mission.

Figure 12 shows the point table values for emitters at a specific time during the mission. The DSS uses these values to control the scan schedule. Mission events cause the point table to change. The DSS must then adjust the scan schedule. The DSS thus optimizes the expected total point value for when the point table is updated.

The assignment of the values in the point table requires expertise in EW mission planning. The mission planning process was not developed in the ANTS program. A complete DSS would contain two levels: Mission Planner (MP) which updates the point table and Dynamic Scan Schedule (DSS) which generates the schedule.

FIGURE 12 Evaulation Metrics – Early Detection Scoring & Max. Scores

Evaluation Metrics - Early Detection Scoring & Max. Scores
(The 5 Lethal emitters A-E vary over the five test spans)

When Detected:	First	Second	Third	Fourth ++ Occurrence
5 Lethal Emitters				
A	2000	1800	1500	-10000
B	2000	1800	1500	-10000
C	2000	1800	1500	-10000
D	2000	1800	1500	-10000
E	2000	1800	1500	-10000
116 Non-Lethal Emitters				
6	334	232	116	0
7	333	231	115	0
8	332	230	114	0
9	331	229	113	0
10	330	228	112	0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
121	233	116	1	0

Max. Span Score <= 43,698 * 5 spans = **218,490 => Max. Mission Score**

Scan Schedule Results

This section presents the results of the DSS tests. The first test compared wide beams and narrow beams. Wide beams showed little difference between FSS and DSS (See Figure 13). However, when the beam width was narrow, the SRI DSS showed improvement over the FSS.

The radars do not actually change their beam width. However, as the receiver gets more distant from the radar, the effective beam width (for detection) gets narrower because the signal gets weaker. This decreases the time that the signal is above the receiver detection threshold. As seen from the figure, narrow beams place more stress on the SS process. This causes greater differences in the DSS and FSS scores.

Note that the SRI DSS caught all 25 lethal emitters while the FSS and NEU DSS missed lethal emitters if the beams are narrow. It is important to note that missing a lethal emitter can cause failure of the mission or loss of the plane.

FIGURE 13 Narrow Beam Widths Stress the System More Than Wide Beam Widths

Narrow Beam Widths Stress the System More Than Wide Beam Widths

Wide Beam Widths and Min. Dwell Durations					set 8
Lethal	**1**	**2**	**3**	**Missed**	**Total**
FSS	24	0	0	1	25
NEU	22	1	0	2	25
SRI	23	1	0	1	25

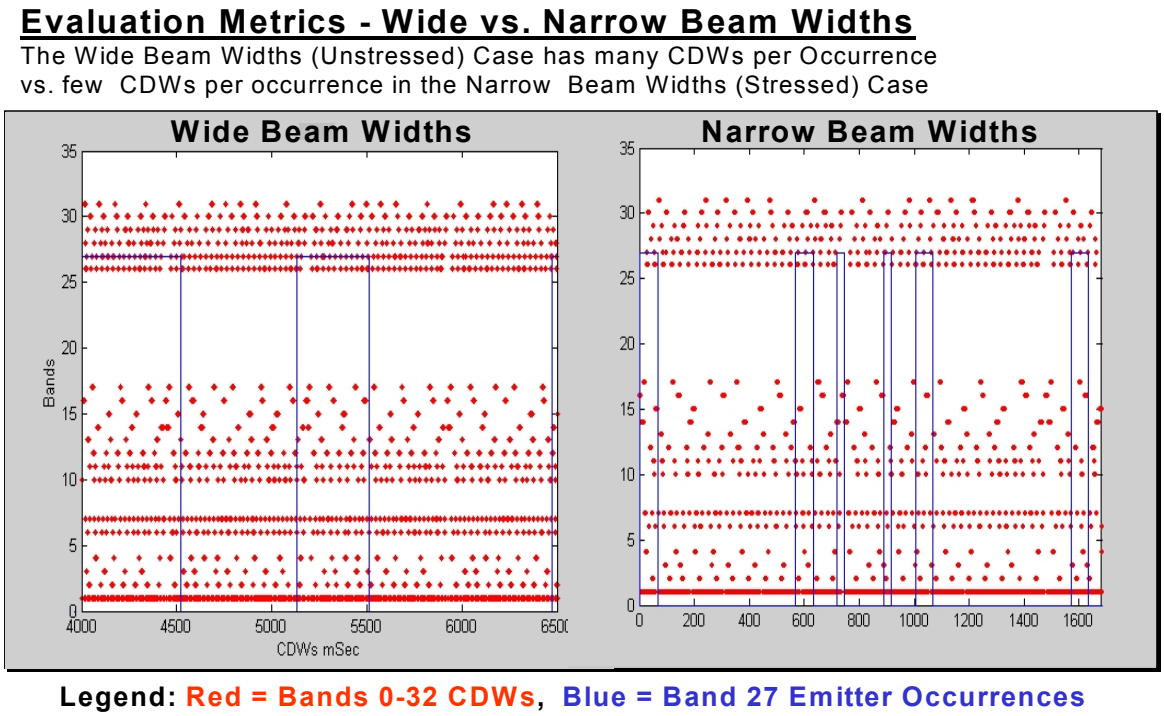
Non-Lethal	**Detects**	**Missed**	**Total Events**	**Total Score**
FSS	561	19	580	197247
NEU	536	44	580	178473
SRI	561	19	580	197408

Narrow Beam Widths and Min. Dwell Durations					set 17
Lethal	**1**	**2**	**3**	**Missed**	**Total**
FSS	22	1	0	2	25
NEU	11	4	2	8	25
SRI	25	0	0	0	25

Non-Lethal	**Detects**	**Missed**	**Total Events**	**Total Score**
FSS	468	112	580	154,037
NEU	438	142	580	52,225
SRI	500	80	580	183,985

Figure 14 shows a timing diagram that compares wide and narrow beams. The EW receiver dwells are shown in red. The blue curve shows a band 27 emitter. If the beam is wide, the emitter is on for a longer time and is easier to detect. Longer revisit periods can be used. This makes it easier to produce a scan schedule. As shown in figure 13, the DSS offers no advantage over a FSS for wide beams. This is because the receiver revisit time can be longer. A longer revisit time makes the receiver duty cycle needed to detect the emitters lower.

FIGURE 14 Wide and Narrow Beams



Figures 15 and 16 shows results as a function of dwell duration required to detect the radars. Radars having long PRIs require longer EW receiver dwells. This requires more receiver duty cycle and makes it more difficult to produce a scan schedule. The most stressing case is narrow beams and long radar PRI.

FIGURE 15 Effect of Longer Radar PRIs

Narrow Beam Widths with Increasingly Longer Dwell Durations Stress the System

Narrow Beams and Min. Dwell Durations				DD=100	set 15
Lethal	**1**	**2**	**3**	**Missed**	**Total**
FSS	25	0	0	0	25
NEU	21	2	1	1	25
SRI	25	0	0	0	25
Non-Lethal	**Detects**	**Missed**	**Total Events**	**Total Score**	
FSS	580	0	580	217,330	
NEU	532	48	580	174,246	
SRI	575	5	580	216,242	

Narrow Beams and Min. Dwell Durations				DD=175	set 16
Lethal	**1**	**2**	**3**	**Missed**	**Total**
FSS	24	0	0	1	25
NEU	15	1	3	6	25
SRI	24	1	0	0	25
Non-Lethal	**Detects**	**Missed**	**Total Events**	**Total Score**	
FSS	539	41	580	184,966	
NEU	427	153	580	76,803	
SRI	533	47	580	192,467	

Narrow Beams and Min. Dwell Durations				DD=200	set 17
Lethal	**1**	**2**	**3**	**Missed**	**Total**
FSS	22	1	0	2	25
NEU	11	4	2	8	25
SRI	25	0	0	0	25
Non-Lethal	**Detects**	**Missed**	**Total Events**	**Total Score**	
FSS	468	112	580	154,037	
NEU	438	142	580	52,225	
SRI	500	80	580	183,985	

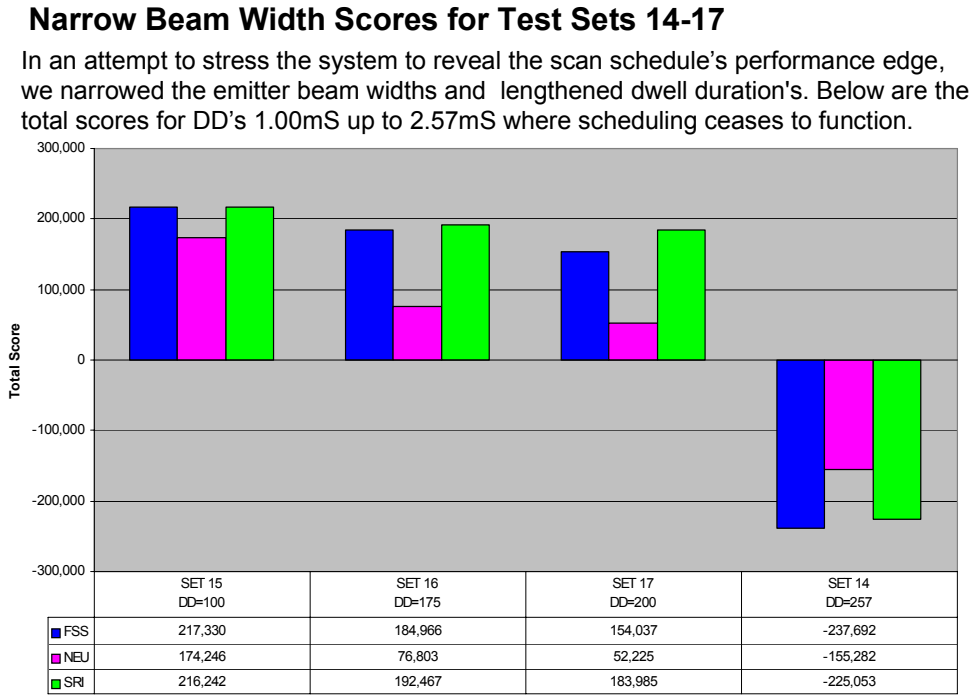
In an attempt to stress the system to reveal the scan schedule's performance edge, we increasingly narrow the emitter beam widths & lengthen dwell duration's.

We used dwell duration's from 1mS to 2.57mS & found the SRI Scan Scheduler performed best at dwell duration's around 1mS but scheduling was limited to less than 2.58mS.

We noted that NEU was better for tracking & that SRI was better for initial detection.

Figure 16 shows that if dwell duration needed is too narrow both the FSS and DSS fail.

FIGURE 16 Narrow Beam Width Scores for Test Sets 14-17



This rapid fall off as the difficulty of the SS problem increased occurs when the duty cycle of the EW receiver is about 90%. SRI found that their SS generation process took longer to produce at SS at high loads. The rapid fall off problem can be addressed by dropping some non-lethal emitters from the schedule or accepting less than 100% coverage of some emitters.

Challenge Problem

Advanced ECM technology use specialized and expensive hardware, realistic stimulation/evaluation environments and secure processing/documentation. These resources are not readily available to most of the ANTs research community. The objective of the Challenge Problem effort is to specify, design and implement an ECM-like challenge problem that is inexpensive to duplicate, unclassified and includes performance measures to enable evaluation of challenge problem solutions. This will allow experimentation of a common problem by the ANTS community and will enable the evaluation and comparison of their solutions produced by the program.

Under the ANTS program BAE designed and built a CP testbed. This testbed consisted of a set of nodes. Each node is controlled by a Linex based PC. Control software for the PC was also developed. BAE defined the API as well as details on the operation of the node. Also, BAE assisted AFRL in the development of a software simulator called RADSIM. RADSIM was used by the ANTS PIs test their approaches. Finally, BAE assisted MITRE in setting up a set of nodes, which were used to conduct demonstrations of various approaches developed by the ANTS PIs

Challenge Problem Metrics

The evaluation metrics and experimental framework to for the challenge problem were selected to exercise pertinent aspects of the approaches taken by the different projects. The metrics for the challenge problem (CP) must be determined by a set of top level goals and should be applicable to all ANTS contractors who undertake the challenge problem. A common set of metrics will allow different approaches to be compared. Since there is no common definition of "agents", the metrics must not be based on specific agent frameworks or approaches. Instead the metrics must be based of the goals defined for the CP "game". This game is to detect and track the target using the following "limited" resources:

Radar sensor – limit the number of "beam-seconds"

Comm link – limit the number of messages

CPU/processor – effective execution of resource allocation schemes and track processing

A good solution will track the target with the lowest possible RMS. To accomplish the tracking, the above resources must be managed. The resource limitations make the CP a resource management problem. The processing is limited by the PC CPU, the radar can be turned on 100% of the time (but not all beams monitored), and the comm link can be on 100% (however, units must not jam each other). Even with maximum possible use of resources, the resource management process must still make decisions (select which

beam, measurement type, and comm link management). To add additional challenge, limits to the use of the above resources can be added. Thus, limits to comm message, radar beam seconds, and CPU loading will be imposed. The processing (CPU loading) metric will include the added CPU load for the resource allocation scheme. The top level system metrics consist of tracker RMS, Radar Beam seconds, Comm link loading, and CPU loading.

A common set of goals selected for the CP game is to detect and track a target using distributed processing and sensing resources, which are subject to limits, is communication, sensing, and processing. Due to limitations in energy, sensing ability, communication, and processing ability of the system, all of the system resources must be effectively managed. It is the proper management of these resources that define a "good" negotiation process. The resource allocation process will optimize the target tracking process. The CP experiments will be set up so that effective management of the system resources (beam time, message traffic, and processor use) is needed to detect and track a moving target. A top-level common set of metrics will measure the overall system performance. The set of system level metrics is directly related to a well-defined set of CP objectives. The features of the CP game (defined by a set of objectives) that make it interesting as a resource management problem are:

Coordinated activity of physically distributed resources is needed to collect and process the data.

The processing for the resource allocation function can be distributed over different physical processors.

The physical distribution forces one to address working with low bandwidth and unreliable communication.

Processing and communication resources must managed effectively.

Sensing resources must be managed effectively.

System scalability is required--more nodes can be added.

The CP involves distributed control (i.e. the software is distributed) of distributed resources.

The resource control schemes used in centralized systems or tightly coupled systems might not form an effective solution (especially if the solution is to be scaled up to many nodes) for the challenge problem due to limitations of communication and processing. The CP nodes are subject to energy constraints, processing limitations, and communication limitations. Under these limitations, the distributed CP node resources must be coordinated to accomplish the tracking task. Since a single node is unable to collect enough data to track the target, a set of cooperating nodes is needed.

The present testbed contains five nodes each having a PC and RF comm link. Four of the nodes will have radar sensors. Note that more nodes can always be added to cover a larger area. The nature of the radar sensor requires coordination of multiple nodes to track the target. Nodes must be able to coordinate their activity and exchange data. This testbed will be used to demonstrate the software that is used to manage this collection of distributed resources.

Solution Independent Metrics

To evaluate the software, a common set of metrics is needed. The metrics must not be specific to the software solution details. Instead, the metrics must relate to a set of CP top level goals. The set of CP goals is:

Have high target detection probability and low probability of false alarm.

Track the target with low error.

Reduce total value of "Beam-Seconds"

Minimize the number of comm link messages.

Scalable – Processing load on any single processor should be independent of total number of nodes.

Fault tolerant - If node processor fails, other node can take over.

Efficient processing – Projected performance must allow use of low power processing. (i.e. Mflops not Gflops)

The evaluation of goals 1-4 requires on-line logging. During a "run" the following metrics will be logged:

Detection events (correct, missed, false alarms)

Track error. (RMS of the track)

Beam-seconds (T= (one beam on) +2*(two beams on time) +3*(three beam on))

Comm link message counters

These quantities will be logged automatically for each node. Goals for beam-seconds and comm link use will be set. A good solution will meet these goals and have a low track RMS. It will not exceed the allocations for beam seconds or comm link message

count. In addition, the approach will be scaleable and make efficient use of the processing resources.

Goal 5 is related to the architecture of the system. Since each node brings its own processor, the addition of nodes does not have to increase the processing load of other processors if processing can be done "locally". A scaleable solution will take advantage of the local processing ability associated with each node. The logged metrics (comm link messages) will be a factor in the evaluation of scalability. Other factors will depend on how the software is distributed. The bottom line metric is the peak processing load on each processor that contains any processing associated with resource allocation or tracking processing. The peak value will be independent of the number of nodes if the approach is scalable.

To be fault tolerant, the system must not depend on any single node. Since the hardware is the same for all nodes, it is not a limiting factor. (Except for the node that displays the result for monitoring). To show complete fault tolerance, the processing (except for the display monitoring) should be able to be accomplished by any of the five nodes.

Goal 7 deals with processing efficiency. The solutions should be effective in terms of processor resources. With N nodes, there will be N processors each having a specific amount of memory and throughput (MIPS). To evaluate the processing load, detailed metrics may be required. Each approach might require some unique metrics that must be logged. These results can then be used to come up with an overall "processing effectiveness" metric that allows a system level evaluation.

To support the different ANTS Challenge Problem researchers a means of profiling/monitoring the executable software is needed. The University of Kansas developed a profiler that was used along with the BAE CP software. The profiler to collects the data that is required. This data was used to evaluate the high level metric listed above.

ANTS Challenge Problem Test Bed

The Challenge problem consists of a distributed micro-sensor array composed of many physically distributed units that sense, process, communicate, and coordinate. Each unit has a sensor, processor, power source, and communication link. The array of units can detect, track, and possibly classify moving objects. Processing and control algorithms are resident in each unit in the sensor array. Distributed resource management/control and sensor data fusion algorithms are required. A centralized process is not desirable because it does not allow the array to scale. In addition, a centralized approach has a single point of failure.

This section discusses the test bed consisting of a set of battery powered micro-Doppler Radars. The section is organized as follows:

Overview

Track Processing,

Test Results and CP benchmark

CP Metrics

Simulation Model for RADSIM

OVERVIEW: Distributed Sensing Test Bed

The Darpa Autonomous Negotiating Teams (ANTS) program is developing approaches for distributed resource management where communication is limited and distributed resources must be coordinated to conduct a task. To develop and test distributed resource management approaches, BAE built a set of CW Doppler radar sensors. Each sensor node consists of a three-beam CW radar sensor, processor, communication link, and a battery. The CW radar sensor has three beams that can be switched. The nodes are distributed over the area to be monitored. Data from multiple nodes must be combined to infer the target location and velocity vector. The set of nodes requires a close coordination between a resource management process and a distributed fusion process in order to detect and track moving targets. Following are some characteristics of this distributed system:

(1) A node (by itself) is unable to measure target location or velocity.
(2) Nodes are battery powered and radar on time must be minimized to save power.
(3) The communication links used by the nodes are limited.
(4) Node measurements must be coordinated in time to measure target location.
(5) Nodes must decide in advance what measurements are needed (beam warm up time).
(6) A distributed data fusion process is needed.
(7) The sensor management process requires collaboration between units.
(8) The sensor management process is reactive to the estimated target state.

Figure 17 shows the details of a radar node. The CW radar emits a 9.35 Ghz radar signal. The reflected signal is mixed with a sample of the transmitted signal and the output of the mixer is sampled. If any object (target) in the beam moves, the output of the mixer changes. When the round trip distance to the target changes by one wavelength, the mixer output goes through one cycle. If the target moves at a uniform speed (v) toward or away from the radar node, the mixer output will be have a frequency that is proportional to the speed. For the 9.35 Ghz radar, f=18.9v where v is in feet/second. The amplitude of the signal is given by $AMP(R,\theta)=k*Exp(-(\theta/w)^2)/R^2$. The value of k depends on the target radar cross-section. The value of w defines the beam width, R is the distance, and θ is the angle from the beam centerline to the target line of bearing.

Node Functions:

1. Sensing
2. Data processing (fusion)
3. Metric evaluation
4. Resource allocation
5. Display

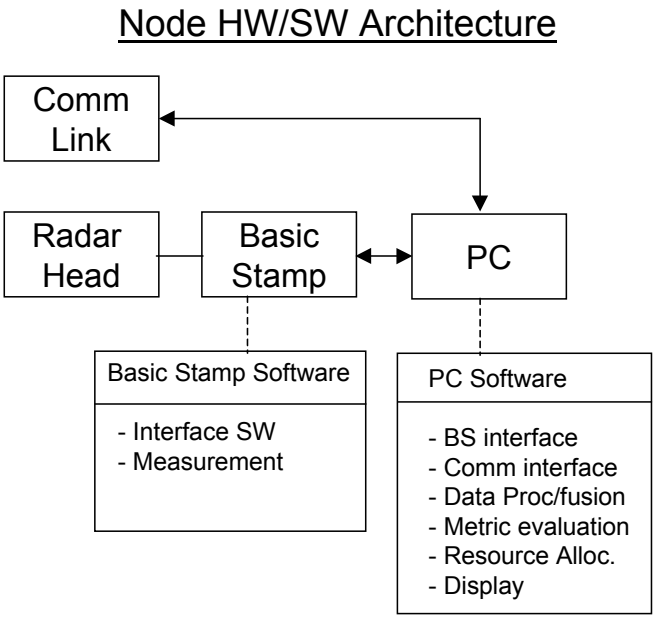


Figure 17 Remote CW Doppler Node

The CW Doppler radar measures the amplitude and frequency of the mixer output. The amplitude data defines the target position (assuming the cross section is known) to be in a set of range and angle values. If the amplitude is A and error is DA, then the target is located in the area defined by:

$$A-DA < AMP(R,\theta-\theta_0) < A+DA$$

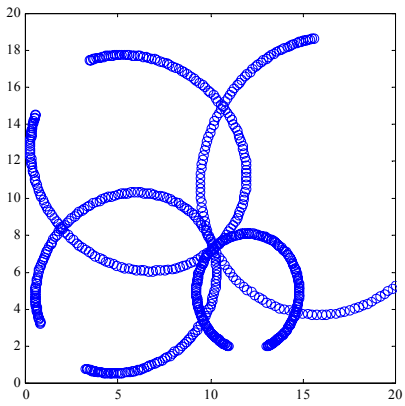


Figure 18. Overlapping Radar Bands

The target can be located in an amplitude band that has ranges and angles consistent with the measured amplitude. Figure 18 shows four overlapping amplitude bands for four nodes. To locate and track targets requires the coordination of multiple radar units. The problem is complicated by the fact that this distributed sensing, resource management,

and data fusion process must be conducted with limited unreliable communication. Sensor power must also be minimized.

Tracking Processing

As stated, a given sensor provides target location to within an ellipse band. A single node also provides one component of the target velocity vector. This component is along the radial from the sensor to the target. There is a sign ambiguity for a single Doppler velocity measurement as well. If the target is detected in two of more beams, it is possible to compute the location and the 2-D velocity vector. The nodes are set up so that it is possible to make two of more simultaneous measurements of the target. In this case, the target location is given by the minimum chisq value. The velocity can also be computed since two measurements are made from different locations. In many cases, the measurements can not always be made simultaneously. Often a single measurement is made. Single measurements can still be used to update the estimated target state. This is done by estimating the target state at the time of the single measurement. Predictions for a complete measurement set can then be made and the measured result(s) can be blended in. Figure 19 shows a block diagram of the tracking/fusion process. There are four major components:

(1) Target location estimator – uses a chisq process to find a location consistent with the measured/predicted set of amplitudes.
(2) Target velocity estimator—uses a linear least squares process to compute the targets 2-D velocity vector from measured/predicted Doppler frequency measurements.
(3) Tracker – A conventional tracker. Predicts target position for measurement blending.
(4) Measurement predictor – uses tracker prediction to generate expected measurements.

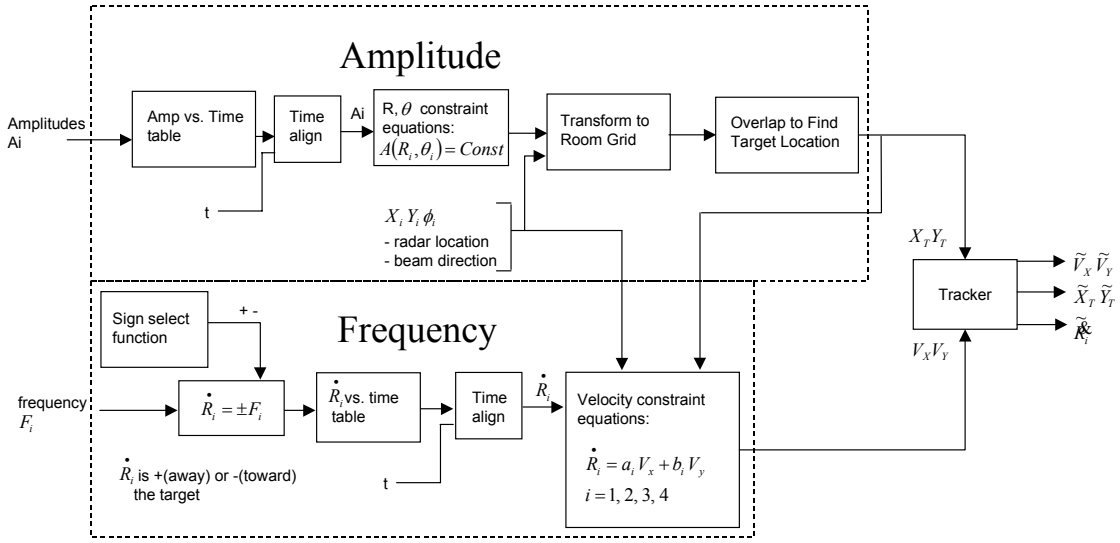


Figure 19. Tracking and Fusion Process

Resource Management Process

The resource manager(s) can do the following local control functions: (1) turn beam on/off, (2) allocate the A/D to a beam, (3) select the type of measurement. The resource

managers must also coordinate activities on order to produce complementary measurements, allocate beams for detection or measurement, prevent any one node from being over used (battery will wear out), assign local processors to do fusion processing, and coordinate node to node communication.

The sensor array consists of a set of N nodes. Each node has the following resources:

(1) Limited Beam-seconds (consumable resource)
(2) Three beam radar which share one A/D channel – one beam can be monitored as a time.
(3) An 8 channel 2 way communication link which can transmit or receive – shared by all nodes
(4) Processor used for fusion/track processing, resource allocation, and communication control.

To accomplish the detection and tracking task, the above resources must be used for the following sub-functions:

(1) Measure background level. (Beam-seconds, A/D)
(2) Detection of targets (Beam-seconds, A/D)
(3) Measure amplitude and/or frequency (Beam-seconds, A/D)
(4) Exchange sensor data, tracker data, or control information (comm link, comm channel)
(5) Conduct the fusion processing (processor)
(6) Conduct the resource management processing (processor)

Each radar node has three beams that are spaced 120 degrees apart. This provides 360-degree coverage as well as a crude resolution of target bearing. The beams can be switched on or off to save power, however, it takes about one second for a beam to power up. The resource management process must anticipate when the beam is needed. It is possible to have 0, 1, 2, or 3 beams powered up but only one beam can be monitored. The radar can be commanded to take different types of measurements: FFT for high resolution Doppler, short dwell for close/fast moving targets, long dwell for distant/slow targets. Each radar node is independently controlled by its own processor. The resource control process in each node must coordinate the measurement activity in order to collect the most optimal measurement set. The process is reactive to the environment. To optimize the sensor data collection the resource managers must consider the following:

(1) The measurements of the highest value are those that reduce the targets uncertainty the most.
(2) Measurements made at closely spaced times from different nodes are highly valuable.
(3) Turning off the beam can reduce power if the target location (present and projected) is out of range.

The sensor control functions need a quality function to determine what to measure. This quality function is defined by the reduction of the target uncertainty produced by the measurement or set of measurements. Note that multiple measurements made at close time spacing are highly desirable because multiple measurements are needed to define a unique target location (see figure 2). However, the quality for a multiple measurement set drops as the time difference between the measurements increases. The factor to account for this is Exp(-abs(t1-t2)*v/d). Here t1 and t2 are the measurement times, v is the target speed, and d is the resolution distance. For the test bed, v is about 1-3 feet/second and d is 3 feet. This implies that time difference must be less that 3 seconds to benefit from measurement simultainty. In addition, a quality factor velocity measurement is computed. Each resource management process associated with a given node (referred to as an agent) can use the following quality evaluation functions:

(1) Single measurement quality (short, long)
(2) Coordinated measurement with another node (short, long)

The "agents" use the quality functions to evaluate possible sets of measurements. The sensor control functions (Agents) in each node work together to select the optimal set of measurements. A joint effort is needed because of the benefit of obtaining multiple measurement sets. To make a high value measurement set two or more agents must agree to allocate the needed radar beams at the proper time.

For single measurements, the quality function is given by:

$U= [AMP(R,\theta-\theta_0)/AMP(2,0)]^{3/2}$ if $AMP(R,\theta-\theta_0) <= AMP(2,0)$

$U= 1.0$ otherwise

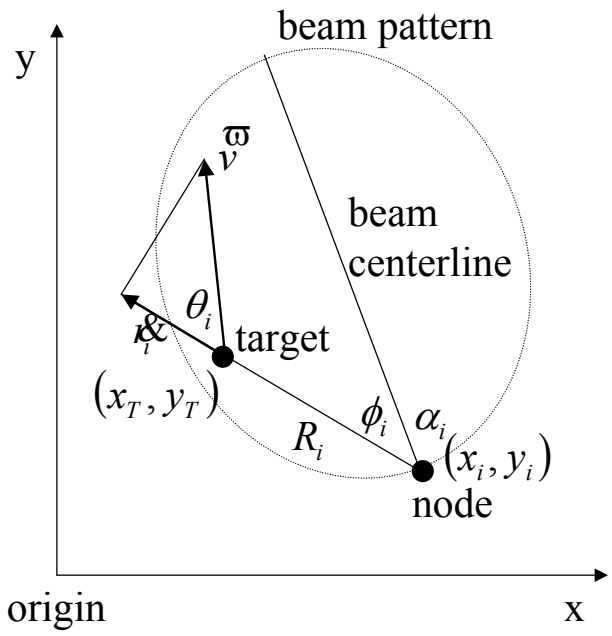
For multiple measurements, the quality function is the sum of each individual quality function (Ui) plus a joint contribution J.

$Um= U1 + U2 + .. UN + J$

In the above, U1, U2.. , UN are the single measurement terms for each node and J is given by:

$J= N*exp(-dt*v/d)$

Note that the assumed target position is needed to compute the above quality values. In addition, the locations and beam direction for each node is needed. Figure 20 defines this geometry.



- *vbar* is target's velocity vector (ft/s)
- $rdot_i$ is radial velocity (ft/s)
- (x_T,y_T) is target's location (ft from origin)
- (x_i,y_i) is node's location
- R_i is distance between target and node (ft)
- θ_i is target's orientation w.r.t. line from node to target (deg)
- ϕ_i is angle between target and active beam centerline (deg)
- α_i is orientation of beam (deg)

Figure 20. Geometry Definitions

During operation, the resource managers for each node must compute these quality functions and use the results to allocate resources. To make a measurement, the beam must first be powered up. Since the radar beams must be power managed and take 1.0 second to power up, the quality of measurements projected in the future is needed. To produce these values, the resource manager specifies the time that the proposed measurement set will take place. The quality function uses the tracker state vector to generate the target location for the proposed measurement time. Note that is some cases, default values are used to evaluate the quality metrics because the targets location may be unknown. This is the case just after the target is detected.

Challenge Problem Constraints and Metrics

The objective of the challenge problem is to provide a test bed for the ANTS program. Since the objectives of the ANTS program is to address highly decentralized and autonomous negotiation of tasks, roles, and allocations for distributed problems, a problem that requires distributed control of distributed resources was selected. The problem involves an array of distributed sensors that must coordinate in order to track objects. The resources to be managed are Radar beam seconds, communication channel, radar pre-processor and node processor. This section discussed how the challenge problem is be used as a test bed. The discussion outlines the constraints and requirements that must be satisfied in order to be successful at the CP game. In addition, the need for a common set of software evaluation metrics that apply to all software solutions is identified.

The software metrics for the challenge problem (CP) must be determined by a set of top level goals and should be applicable to all ANTS contractors who undertake the challenge problem. A common set of metrics will allow different approaches to be compared. Since there is no common definition of "agents", the metrics must not be based on specific agent frameworks or approaches. To the maximum extent possible, the metrics chosen for the challenge problem should be as equally valid in the Rome Lab's simulator as they are in the real hardware.

Before defining a common set of metrics, the "constraints and requirements" of the challenge problem must be defined. These constraints must be satisfied and are based on goals defined for the CP "game". The CP game is to detect and track the target using the following "limited" and distributed resources:

Radar sensor – limit the number of "beam-seconds"

Comm link – limit the number of messages

CPU/processor – effective execution of resource allocation schemes and track processing

A good solution will track the target with the lowest possible RMS. To accomplish the tracking, the above resources must be managed. The resource limitations make the CP a

resource management problem. The processing is limited by the PC CPU, the radar can be turned on 100% of the time (but not all beams monitored), and the comm link can be on 100% (however, units must not jam each other). Even with maximum possible use of resources, the resource management process must still make decisions (select which beam, measurement type, and comm link management). To add additional challenge, limits to the use of the above resources can be added. Thus, limits to comm message, radar beam seconds, and CPU loading will be imposed. The processing (CPU loading) constraint will include the added CPU load for the resource allocation scheme. The constraints are tracker RMS, Radar Beam seconds, Comm link loading, and CPU loading. More difficult Challenge Problems can be made by "tightening" the constraints. Other requirements are to address problems where the number of sensor nodes is scaled up. Approaches that employ centralized resource management or make use of extensive communication between nodes do not satisfy the scalability requirement of the CP. To meet the scalability requirement, one possible approach might be to have distributed control agents that have a low rate communication. Note, however, communication between agents in the same physical processor is not as costly as communication between different nodes. Extensive communication between nodes might limit the system to scale because the communication channel will saturate.

An issue in selecting the software metrics is to what extent is the set of metrics dependent on the negotiation (or other resource allocation) approach. Since it is the objective to evaluate and compare different approaches, a set of metrics that is common to all approaches must be used. Developer specific metrics will only be of use by the developer for optimization. These method specific metrics can not be used to compare methods. In particular, if approach A shows a high score for a "method A specific" metric, but has poor performance with respect to the common set of "CP game" requirements, the metric is useless for comparison of approaches. Measurement of a metric (such as number of negotiation rounds) is useless if the negotiation process does not satisfy some common system level goals. The only way to compare different approaches is to have a common set of metrics and insure that the overall performance meets the CP constraints and requirements. It is not possible to compare the effectiveness of different negotiation or resource allocation schemes unless one defines a common "goal" that is to be satisfied.

A common set of goals selected for the CP game is to detect and track a target using distributed processing and sensing resources that are subject to limits is communication, sensing, and processing. Due to limitations in energy, sensing ability, communication, and processing ability of the system, all of the system resources must be effectively managed. It is the proper management of these resources that define a "problem effective" negotiation process. The resource allocation process must optimize the target tracking process subject to CP constraints. The CP experiments will be set up so that effective management of the system resources (beam time, message traffic, and processor use) is needed to detect and track a moving target. A common set of constraints and

requirements is defined for the CP. The features of the CP game (defined by a set of objectives) that make it interesting as a resource management problem are:

The processing for the resource allocation function should be distributed over different physical processors.

The physical distribution forces one to address working with low bandwidth and unreliable communication.

Processing and communication resources must managed effectively.

Sensing resources must be managed effectively.

System scalability is required--more nodes can be added without needing faster processors or more communication capacity.

The CP involves distributed control (i.e. the software is distributed) of distributed resources.

The resource control schemes used in centralized systems or tightly coupled systems might not form an effective solution (especially if the solution is to be scaled up to many nodes) for the challenge problem due to limitations of communication and processing. The CP nodes are subject to energy constraints, processing limitations, and communication limitations. Under these limitations, the distributed CP node resources must be coordinated to accomplish the tracking task. Since a single node is unable to collect enough data to track the target, a set of cooperating nodes is needed.

The present testbed contains five nodes each having a PC and RF comm link. Four of the nodes will have radar sensors. Note that more nodes can always be added to cover a larger area. The nature of the radar sensor requires coordination of multiple nodes to track the target. Nodes must be able to coordinate their activity and exchange data. This testbed will be used to demonstrate the software that is used to manage this collection of distributed resources.

Solution Independent Metrics

To evaluate the software, a common set of metrics is needed. The metrics must not be specific to the software solution details. Instead, the metrics must relate to a set of CP top level goals. The set of CP goals is:

Have high target detection probability and low probability of false alarm.

Track the target with low error.

Reduce total value of “Beam-Seconds”

Minimize the number of comm link messages.

Scalable – Processing load on any single processor should be independent of total number of nodes.

Fault tolerant - If node processor fails, other node can take over.

Efficient processing – Projected performance must allow use of low power processing. (i.e. Mflops not Gflops)

The evaluation of these goals requires on-line logging. During a "run" the following metrics will be logged:

Detection events (correct, missed, false alarms)

Track error. (RMS of the track)

Beam-seconds (T= (one beam on) +2*(two beams on time) +3*(three beam on))

Comm link message counters

These quantities will be logged automatically for each node. Goals for beam-seconds and comm link use will be set. A good solution will meet these goals and have a low track RMS. It will not exceed the allocations for beam seconds or comm link message count. In addition, the approach will be scaleable and make efficient use of the processing resources.

Scalability is related to the architecture of the system. Since each node brings its own processor, the addition of nodes does not have to increase the processing load of other processors if processing can be done "locally". A scaleable solution will take advantage of the local processing ability associated with each node. The logged metrics (comm link messages) will be a factor in the evaluation of scalability. Other factors will depend on how the software is distributed. The bottom line metric is the peak processing load on each processor that contains any processing associated with resource allocation or tracking processing. The peak value will be independent of the number of nodes if the approach is scalable.

To be fault tolerant, the system must not depend on any single node. Since the hardware is the same for all nodes, it is not a limiting factor. (Except for the node that displays the result for monitoring). To show complete fault tolerance, the processing (except for the display monitoring) should be able to be accomplished by any of the five nodes.

Goal 7 deals with processing efficiency. The solutions should be effective in terms of processor resources. With N nodes, there will be N processors each having a specific amount of memory and throughput (MIPS). To evaluate the processing load, detailed metrics may be required. Each approach might require some unique metrics that must be

logged. These results can then be used to come up with an overall "processing effectiveness" metric that allows a system level evaluation.

The above discussion deals more with the CP requirements and constraints. In the event that the above CP constraints are not satisfied, they can then be considered as metrics. Note, however, they reflect how the software accomplishes the CP goals rather than a general evaluation of software methods. Other metrics to evaluate software methods in general need to be developed.

Solution Dependent Software/Algorithm Metrics

Details of the different software or algorithm solutions may require specific software metrics. These metrics will not be consistent from implementation to implementation and it will be the responsibility of the individual ANTS Challenge Problem researcher to define and collect their individual metrics. For some solutions these metrics might include number of agent to agent messages or number of negotiations. Other solutions will require different metrics. If the approach does not perform well according to the detailed metrics, it may (or may not) not perform well as judged by the CP requirements. The purpose of the detailed metrics is for optimization of the specific solution. They provide a detailed insight. They also may be used as part of the resource optimization process. Finally, if the user solution is not run on the CP CPUs, a method of estimating the CPU loading must be generated by using the user specific metrics.

To support the different ANTS Challenge Problem researchers a means of profiling/monitoring the executable software is needed. Since the challenge problem PC is Linux based, the approach is to provide an off-the-shelf profiler. The users can then set up the profiler to collect the data that is required. Some of this data may be needed to evaluate the high level requirements (5-7) listed above. Note, however, the monitoring tool will only measure the performance of the code that executes in the CP processors. If users have additional processor resources, the added processing must measured by the user and included in the processing metric.

CP Tests

Tests using the CP hardware were conducted. The objective was to refine the test procedure, verify CP hardware operation, and identify potential problems or pitfalls that might face CP users These tests were done by UMASS, NEU and Sanders with help from James Lawton (RADC) and Alex Meng (Mitre).

Hardware/API checkout

Tests using the BAE CP benchmark software were conducted to insure that the comm link and radar sensor were functional. The results showed the following:

(1) The comm links were functioning properly and can be used to exchange timing offset data (needed for successful tracking) and measured amplitude and frequency data. This test was done using the BAE benchmark CP software as well as a special program that exercises the comm link.

(2) The radar sensor produced repeatable data and this data was used by the tracker to produce good tracks. The quality of the track was in fact better than the tracks generated by an earlier CP benchmark test at Mitre. Possible reasons for this are that (a) the target was moving more slowly and (b) the structure of the room (concrete vs. raised floor), and (c) the geometry was different. The figure shows the track loop that was produced using the Sanders CP benchmark in the NEU test.

UMASS testing

The UMASS CP software was also tested. Following are some observations and lessons learned.

(1) A compatibility problem was identified. The UMASS software was using a different version of Java than the Sanders setup. Corrective action for this has been identified. From my understanding, the Java version being used by UMASS is faster than the version that Sanders was using and was required by UMASS because of performance issues. The lesson learned is that version chances may cause "bug" type problems that must be fixed before the CP testing can start.

(2) The simulation provides a good start at setting up a CP solution. However, the comm link model and sensor model in the simulation can not substitute for the real hardware. For this reason, UMASS will borrow two CP nodes to further debug their solution. I recommended that once they get their interfaces to the hardware working, they collect a file of the inputs they send to the tracker.

(3) The tracker and the resource allocation process form a "feedback" system since the resource allocation must use the tracker output to control the collection of data that feeds the tracker. One challenge (which must be shared between the tracker design and resource allocation) is to solve the initialization problem. The tracker can not provide useful data to the resource control process until it has a good state estimate of the target. At the same time, the resource control process needs a "good" track state in order to cause the data needed for the tracker to be collected. If errors in tracking cause problems in the resource allocation process and these resource allocation errors further degrade tracking, an unstable process will occur. Both the tracker and the resource allocation process must form a "stable" loop.

(4) Since the performance of the tracker/resource allocation loop is data dependent, one can not expect that a simulated node (radar/comm link) can be used to do the final system test unless a sufficiently detailed model of the sensor physics and comm link details is used. It may not be practical to model to this level of detail.

RECOMMENDATIONS and OBSERVATIONS

Initialization Process

Just after the first detection, the tracker has very large errors. The best way to obtain the target location is to observe which beam has the highest amplitude. Assume that the sensors start out by monitoring the beams (detection mode). If an event (detection) occurs, the amplitudes for the other beam can be measured (using a "short time" measurement) in each beam. The beam with the highest amplitude is of most interest. Any other sensors that have beams that intersect this beam (especially close by sensors) should be candidates for data collection. This process should be used until the tracker has collected about 5 measurements. The process of generating crude location by using a amplitude measurement can be considered as part of the tracker. If this is done the first measurement places the target in a "window" that is about +- 50 degrees from the

centerline of the beam with the strongest signal. To obtain a more detailed window for a single amplitude measurement, assume that the error in amplitude is DA. Then compute the "rings" for A+AD and A-DA. The target is between these rings with a probability of about 67% if DA is the one sigma variation of the amplitude measurement. A process that selects between the tracker and the single lobe measurements is needed. Some of the work being done on dynamic composition may help with this selection.

Stability factors

The stability of a "closed loop" tracker/resource allocation process is (in part) related to the amount of data needed by the tracker. The tests done using the CP Sanders benchmark show that the tracker can maintain a good track if data is collected at 1 second intervals from each sensor. The tracker does not require data to be simultaneously obtained from multiple sensors. During the test at NEU, the mean time between measurements from the same sensor was between 0.8 to 2.0 seconds. In addition, the measurements between different sensors were not synchronized. The result is that the mean time between two measurements from different sensors was about 0.25 seconds. However, this 0.25 seconds was not between sensors that had the best beam overlap. Note in the above, the target was moving about 1 ft/second. With measurements taken at these time intervals, a good track (figure shows < 2 foot errors in most cases) can be produced. Better results can be expected if measurements are coordinated to occur in beam overlap regions. To determine if the tracker/resource allocation loop is stable, the performance of both subsystems must be evaluated. We need to know tracker performance as a function of density and distribution of measurement points and resource allocation performance as a function of input data quality. Design of both subsystems must address robustness. Note that the UMASS resource allocation process and the tracker were effective in the simulator (RADSIM). However, RADSIM does not require the tracker to run real time (thus more measurements can be made) and the simulated measurements have less error than the real data.

Recommended Approach for CP tests

(1) The PC "environment" (software—Java versions, PC hardware …etc.) needs to be standardized. Any changes in versions may result in problems that must be debugged before the CP test can start. The BAE CP benchmark should be run to check the configuration. If extensive configuration changes are made, it is recommended that the CP user modify the BAE CP solution to run on their configuration. This will serve as an example and allow the user to gain experience with the system.

(2) The BAE benchmark software should be run before and after tests to identify any hardware failures or any changes in the setup. Possible differences in the Lab (floor, walls--) will be detected by using the bench mark software.

(3) CP users should log the following: (1) measurement file with time (milliseconds), node number, amplitude, and frequency. (2) track file with time, x, y, vx, and vy. Files should be readable in Matlab. The configuration file, track layout file (like the one done by Mitre), and information on the target speed are needed.

RESULTS FROM BENCHMARK CP and UMASS test

Figure 21 shows the amplitude data collected from the four nodes. Each node is plotted using a different color. The horizontal axis goes from 0 to 250 seconds. Note that the time between around the loop is about 50 seconds (the peaks for each node are 50 seconds apart). Since the track is about 10 feet on each side (40 feet around) the speed of the target is about 4/5 feet/second. Also note that the target is always in least one strong radar beam as it travels around the track. This shows that the density and placement of the nodes will provide sufficient track data. The peaks have a region where the amplitude drops to a low value. This occurs when the range rate of the target is zero at the target's closest approach distance.

FIGURE 21 Amplitude Data From Four Nodes

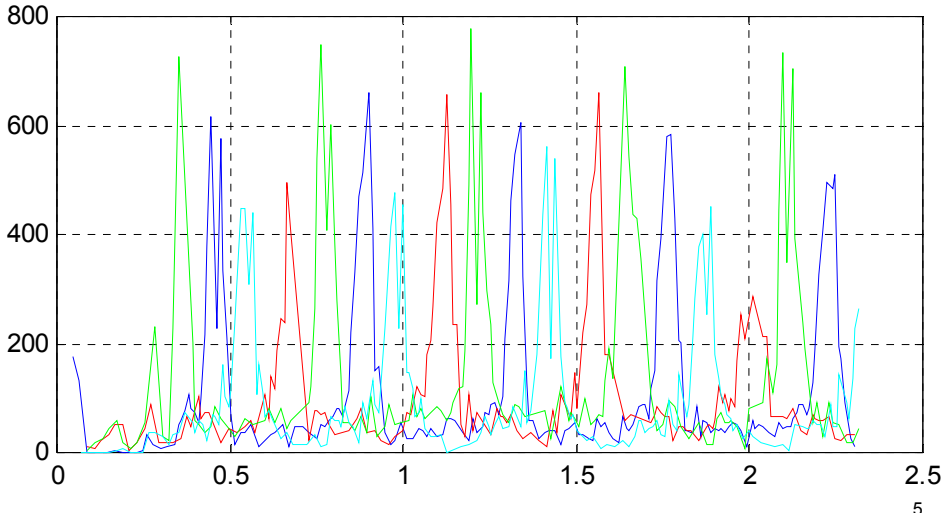
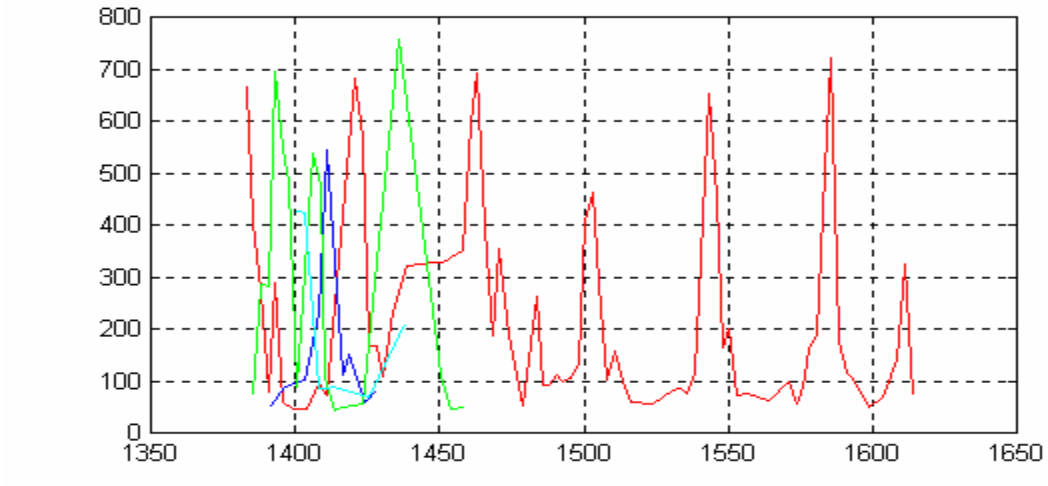


Figure 22 is data collected from the UMASS demo. The red curve is from node 2. The other nodes were not sampled. However the target did make 5 or 6 loops as seen by the peaks in the red curve.

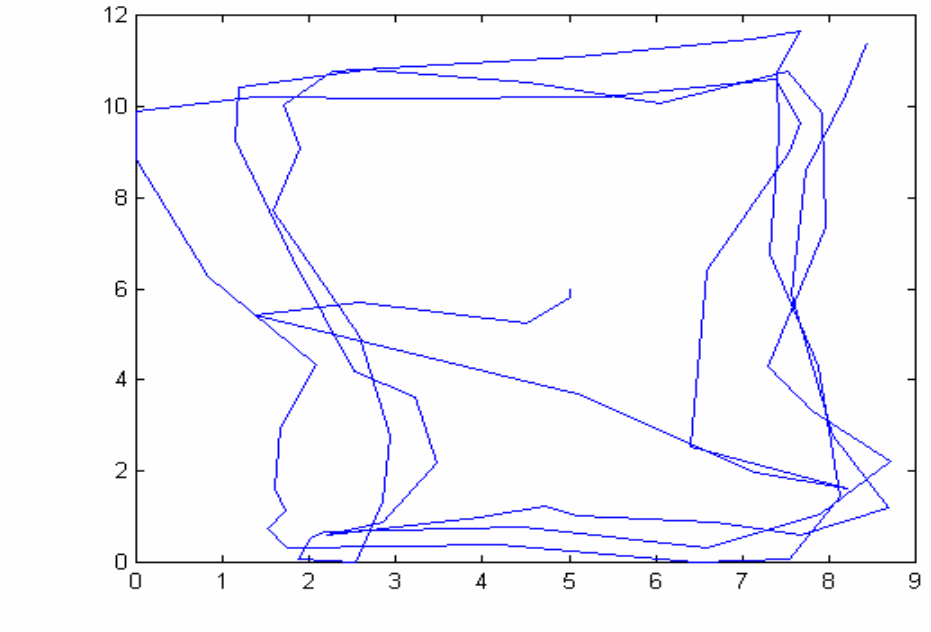
FIGURE 22 Amplitude Data From the UMASS Test



BENCHMARK TRACK

The tracker uses the amplitude data to produce a track. The following figure shows a track produced by the tracker in the Benchmark CP. Note that the initial track estimate had a large error. However after the tracker locks on, the error is much smaller. The target makes 4 trips around the track. The errors are roughly 2-3 feet.

FIGURE 23 Benchmark Track



SECOND UMASS TEST

Figures 24 and 25 are the amplitude and track results from a second test by UMASS at NEU. In this case all nodes were sampled. The total time of the test was 120 seconds and about 40 samples were obtained from each node. Thus the time between samples for each node is 4 seconds. Compared to the Sanders benchmark, the data rate is ¼ of the sanders data rate. The tracker accuracy is expected to degrade. However, the number of beam seconds is reduced. A higher measurement rate is recommended in order to obtain better track performance.

FIGURE 24 Amplitude Data from UMASS Test

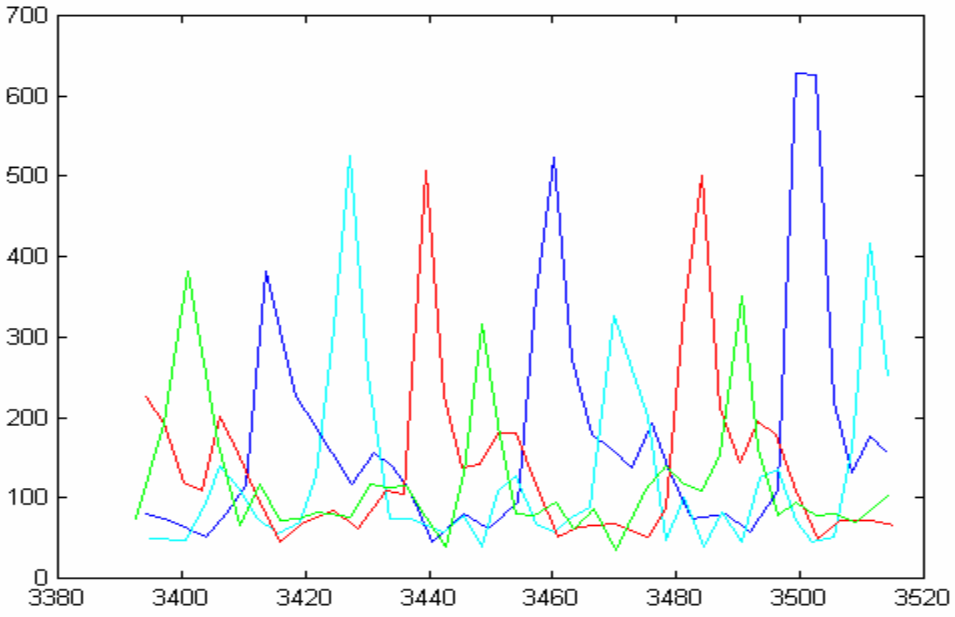
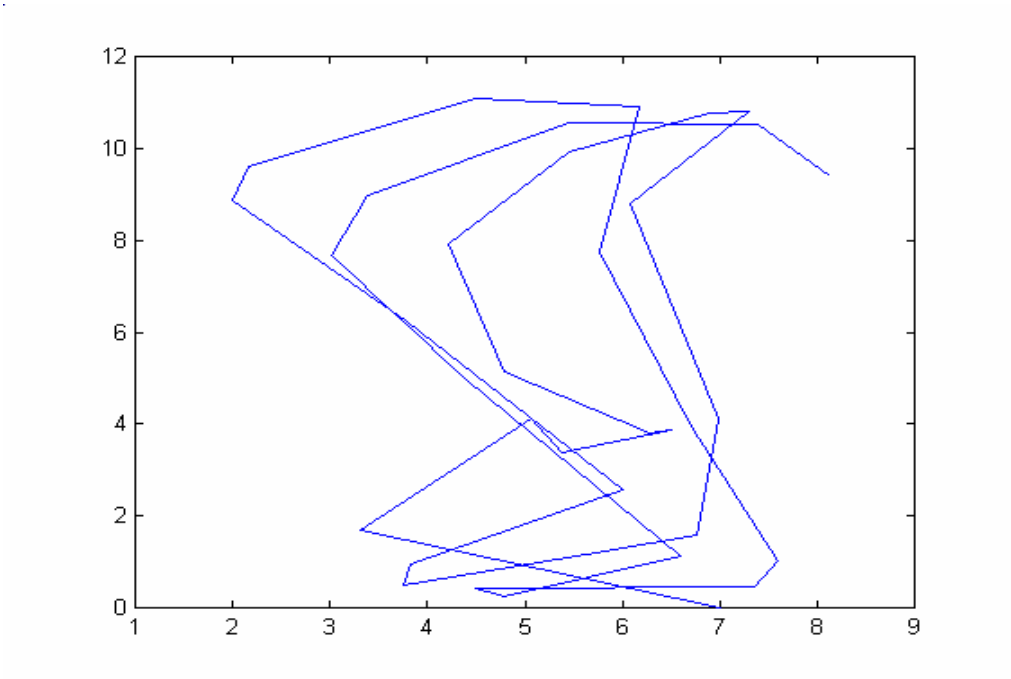


FIGURE 25 Track Data from the UMASS Test



Possible enhancements Identified by the tests

As discussed above, the tracker must be initialized in order to produce good estimates of target location. This is especially true when the output is used to determine the beams to measure. One area that needs improvement is to figure out how to allocate measurements during the initialization process. During the initialization, the data produced by the tracker has large errors. Perhaps an approach that uses the measured beam amplitudes to control resource allocation should be used during the initialization of the tracked.

The tracker also can be improved. The possible areas are:

(1) Each time a measurement is obtained, compute an estimated state vector and measurement set at the time of the measurement. Then update the state using the new (single sensor) measurement. Use the predicted values for the unmeasured beams to reconstruct a complete measurement set. This enhancement can be done using a data set collected from the CP benchmark demo.

(2) Decrease the measurement time. The current CP hardware has choice of three fixed measurement intervals. Possible improvement may be obtained by using shorter intervals. The alternative is to run the target at a slower speed. Note, that there is a limit to how much the measurement time can be decreased. Sanders will investigate this and determine the optimal values.

(3) Recalibrate the sensors. The shape of the beam (assumed in the tracker) and the amplitude coefficient can be more carefully measured.

(4) Measurement weighting. Measurements that are close to the node and in the beam direction have smaller errors. The weighting scheme should reflect the measurement error.

(5) UMASS suggested adding an error window. The window will be big just after detection. It will get smaller after the tracker has locked onto the target.

Lessons Learned

Following are some topics that were brought up at the NEU CP test.

(1) The comm link works for the BAE CP test. However, it may still not be sufficient for a standard Agent based approach. – ANTs CP developers must develop approaches that do not rely on reliable high bandwidth communication. With unlimited communication and processing capability, the solution of the problem is easy.

(2) It was clear that real hardware presented a "different" problem than the one solved on the RASDIM. However, this is the kind of change that must be expected when going from simulation to real hardware..

CP Hardware Model For RADSIM

This section describes a model for the CP hardware. The model describes the major features of the challenge problem hardware. In particular, the model describes the following:

The error in the amplitude measurements is larger than expected. The major source is due to multi-path.

When a target approaches the radar and turns the corner, the amplitude becomes very small even though the target is close. This is due to the frequency response of the mixer.

Multiple targets are expected to be a major "noise" source. The model addresses multiple targets.

Frequency measurements give zero value when the amplitude is small.

Tests at MITRE indicate that there is unit to unit and beam to beam differences in the amplitude response.

Tests at MITRE indicate that the amplitude variation is not described by 1/R^2 where R is the range.

The model described here addresses all of the above items.

RadSim Model

The guiding question is "How much fidelity is appropriate for RadSim". The RadSim model as of August 2001 departed from the hardware in ways that caused the experience of running on the simulator to be different than that of running on the actual hardware in ways that where both fundamental and unfortunate (i.e., not intended). Several modifications to the RadSim model have been made, but the result is still not adequately in line with the hardware.

One solution is to have RadSim directly simulate the low-level measurement processes. The key advantage of this approach is that it is easy to define. Almost no additional experiments need to be preformed, and there is relatively good agreement on the nature of the resulting model.

The other solution is to "fix" the few egregious limitations of the existing model, while trying to maintain a relatively high level of abstraction. The result would provide greater insight and intuition to problem, the changes would be easier to code and test, and use execute far more efficiently..This document presents a high level model that captures key features seen in the real hardware. These features consist of the following:

There are nulls (ampltide=0) when the target is has zero range rate.

The errors in the real hardware are larger than the simulated errors.

The frequency is reported as zero when the amplitude is low.

A description of the CW radar signal is described in the next section. Then, a detailed description of the hardware is presented. This was used to define the RADSIM model.

Background

Each CP sensor consists of three 9.35 Ghz CW radar sensors that are aimed 120 degrees apart. The control computer sends commands to the CP sensor. These commands cause the sensor to make measurements. The command selects the beam and specifies parameters of the measurement. Details of the measurement noise, biases, radar signal, and environment effects are discussed here. The objective is to provide a model of the noise, systematic effects, and environment effects. Note that the model can be either a detailed emulation of the radar or a simplified summary. We first present details of the radar signal to allow better understanding of the limitations of a simpler more abstract model which is recommended for use in RADSIM.

Free Space Model

The free space model is based of basic radar theory and the specification of the radar components used in the CP sensor. The model ignores multipath and motion of objects in the environment.

The 9.35 CW radar emits a RF signal that is on continuously. This signal reflects off objects and travels back to the unit. The unit "mixes" the reflected signal with a sample of the transmitted signal. If all the objects in the environment are stationary, the output

of the mixer is constant (except for thermal noise and amplifier drift). The mixer output is passed through a highpass filter and then sent to the signal processing chain. In a stationary environment, the signal observed by the signal processor is zero (except for thermal noise). If any object which reflects radar moves, the mixer output will change. The rate of change is determined by the speed of the object and the wavelength of the radar. If the distance to the object (i.e. target) changes by ½ wavelength, the mixer output goes through one cycle. The wavelength is 3.2 centimeters. Thus, the frequency of the mixer signal for a target having a range rate of v cm/second is v/1.6 Hz. Note, that if v is very slow, the signal will be filtered out by the highpass filter.

The amplitude of the signal is determined by the targets radar cross section and the distance (R). The radar range equation is P = K/R^4 where K is a constant and P is power. The mixer output is given by 1/R^2 because the mixer output is related to amplitude (A) and A is the square root of P. Deviations from the 1/R^2 law may occur if there is multipath or nonlinear amplifier behavior.

The amplitude of the mixer output also depends on the angle between the beam center line and the target direction. This dependency is given by exp(-a T*T) where a is a constant and T is the angle. Note that this is 2 dimensional model and assumes that the target is at zero elevation relative to the radar beam centerline.

Signal Model in Free Space

The mixer output for a target located at R(t)=sqrt(x(t)^2 + y(t)^2) is given by:

S(t)=K*exp(-a*T*T)*sin(4*pi*R(t)/3.2 + P)/R(t)^2

R(t) the distance from target to the radar, T is the angle between the beam centerline and the target direction, a is the beam width parameter, K is a constant, and P is an arbitrary phase angle. The signal processor obtains a amplitude measurement (G) and frequency measurement (F) using a highpass filtered version of S(t). If SP(t)=Highpass(S(t)), then G is obtained by G=Max(SP(t)) – Min(SP(t)) for t in the measurement window defined by (t , t + dt). The dt is a selectable measurement parameter (see the CP API document). A typical value is .5 second. For a target moving at 32 cm/second directly toward the unit, SP(t) will have a frequency of 20 Hz and will go through 10 cycles during the 0.5 second measurement window. The signal processor obtains F by counting zero crossings of the signal SP(t).

Error and System Model

The errors and systematic effects are determined by the following sources:

Windowing effects

Parameter errors (i.e., beam width a, non 1/R^2, unit variation K)

Environment effects (multipath and moving objects)

Mixer noise.

These sources are discussed in the following sections.

WINDOW RELATED ERRORS

Frequency Error due to window size

The frequency is obtained by counting zero crossings that occur in the sample window (dt). If the count is N, then the reported frequency F is F=N/0.5 Hz. Since N is an integer, the frequency will be in steps of 1.0/dt Hz where dt is the window size in seconds. The reported frequency is a multiple of the step size and will be lower than the actual frequency.

Windowing Error for amplitude

The value of G is supposed to approximate the envelope of S(t) which is 2K/R^2. The error (even if there is no noise or multipath) in G is related to how much R(t) changes during the dt window. The change in R(t) is a function of target speed. For a target with a range rate of 32 cm/second, with the at a 150 cm range, the envelope amplitude changes from K/1.50^2 = .44 K to K/1.34^2 = .56K. Thus, the windowing error at a "close" range is up to 25%. However, if the target moves from 5.00 meters to 4.84 meters the envelope changes by 7.5 %. The windowing error can be modeled by using the change in range dR during the dt window. The unit will tend to overestimate the value of G. This error can be predicted and used to obtain a more accurate estimate. This was not done in the signal processor since other errors will dominate the results.

Parameter Effects

The model presented above assumes parameters for the units. These parameters are discussed in this section.

Beam width parameter (a)

The value of "a" defines the beam width. Note that even though the beams are 120 degrees apart, the width is not +- 60 degrees. The beams do not have "hard" limits. A beam is described by the exponential function exp(-a*T*T). This function is an approximation of the radar beam shape. The value of a is the only free parameter. This value can be adjusted to better represent the beam. One can model the effect of error in "a" by using the "true" value in the simulation and an "estimated" value in the tracker. The simulated error will depend on the difference between the values.

Radial dependency

Radar theory theory predicts 1/R^2. Mitre has made measurements and fitted data to obtain values for the exponent. The results were1.4 to 1.7 instead of 2. Following are some possible reasons for the unexpected results:

Multipath—Large multipath effects were observed.

Fit details—A correlation between the zero level noise and the exponent was seen.

Non linear mixer response

At present, it is unclear why this behavior is observed. A test using a 1/R^1.46 in the tracker should be used to determine if any improvement will result.

Environment effects

Multipath effects are significant. Reflection from the floor can be as strong as the direct path. The direct and reflected signals will interfere with each other and produce nulls and peaks. A model of this can be made using the expression for S(R(t)) discussed above. Consider that the target and radar unit are d/2 centimeters above a reflecting floor. Note that concrete is a good reflector at 9.35 Ghz. To model this a "virtual" target located at a distance d under the real target can be used. Then, the mixer output is given by:

STOTAL(R(t))=S(R(t)) - Q *S(sqrt(R(t)^2+d^2)

The two S functions interfere and the resulting signal ranges from A(1-Q) to A(1+Q). For a perfect reflector (Q=1) the signal can range from 0 to 2A. A change from 0 to 2A occurs when the path difference between the direct and reflected path chances by ¼ wavelength (0.8 cm). To show the effect, a plot of STOTAL vs R was made. The example had the target and radar 25 cm above the floor and assumed Q=70%. The green curve shows the free space results and the blue curve shows the effect of the floor bounce. If the target moves at 32 cm/second and the sample window is ½ second, then the sampleing window is 10 cycles long. Inspection figure 26 shows the magnitude of the window related error. The reported amplitude is based on the minimum and maximum value in the sample window.

FIGURE 26 Raw Mixer Output

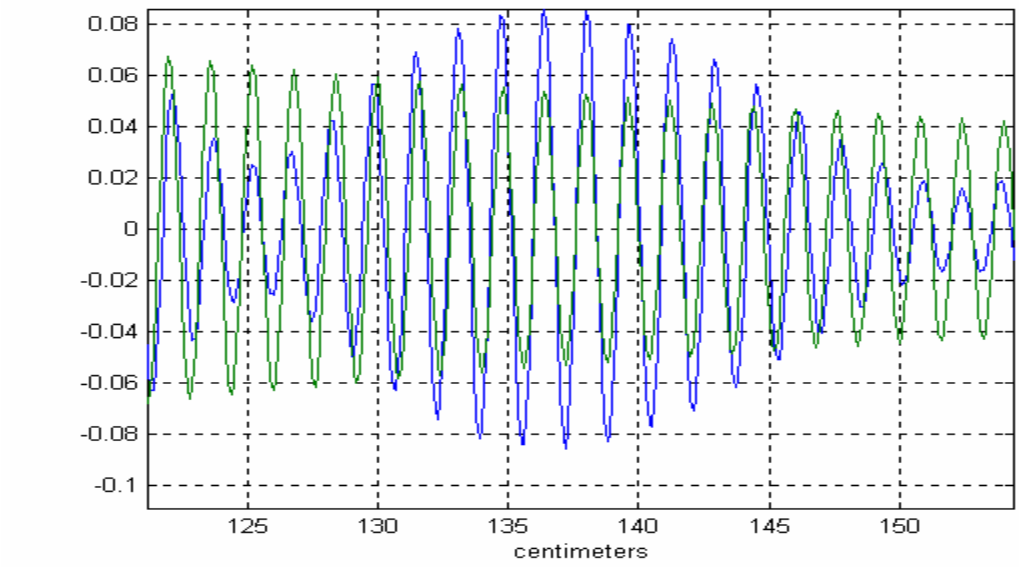


Figure 26 shows that the separation between peak and null is about 10 cm at 120 cm from the target. Note that the window is 10 cycles (or 16 cm long) for our example case. Therefore it will be difficult to use the unit to see the nulls at this close range (120 cm).

The plot also shows the effect of the window. The amplitude changes from .067 to .052 during the window. The multipath can have a significant effect on the measurements. The windowing effect and multipath make calibration of the unit difficult.

Multiple targets and other moving objects

Multiple targets and other moving objects can cause significant errors. One way to model the error is to use the expression for S(R) for each target. The target is at R1 and the interfering target is at R2. Note that R2 may be in a different direction. To obtain a meaningful measurement of the target, the signal from the interfering target must be small (less than 5 or10 %) compared to the main target. This will occur if the interfering target is at least 3 times farther that the main target.

Other moving objects can degrade the measurement results. A person has a radar cross section of 3 times the target. Thus a person can produce a significant disturbance. Since the radar signals easily pass through sheet rock or wooden walls, people in other rooms can cause significant errors in the results.

Signal Noise

The mixer and amplifier also have noise. However, this noise is small compared to the other noise sources discussed here.

Filter Cut Off effects

The mixer is AC coupled to the A/D. Therefore the signal is attenuated for small range rates. This occurs when the target is moving on a tangent or the target is moving slowly. It is possible to have a very low signal level when the target is close to the radar if the range rate is not changing. This occurs at the closest point of approach since this is a stationary point (i.e. a minimum).

Unit to Unit Variation

The value of K includes the effect of the amplifier gain. Unit to unit differences will require different K values. Tests at Mitre show differences in the gain of up to 40 %. Multipath and window effects make it difficult to calibrate the units. The method of choice should determine the K values “in place” and the sensor should not be moved after the calibration. If this is done, the values of K will include unit to unit differences as well as multipath effects.

Model Overview

The reported frequency is N/dt where dt is the length of the sample window in seconds and N is the number of half wavelengths—truncated to an integer. The number N is INTEGER(dR*2/Lambda) where Lambda is the radar wavelength The value dR is change in the target to radar distance R during the measurement window dt. The reported amplitude is multiplied by a function of frequency FCT(F) If the amplitude (after frequency scaling) is less than twice the background noise level then the returned frequency is identically zero. Each node will have a different mapping between amplitude and position. The relation will be,

$$A = \frac{k_0}{r^{\gamma}} \exp\left(-\left(\frac{\theta}{k_1} \right)^2 \right) + \eta$$

where, the constants k_0, k_1, and γ are read from a configuration file and eta is the noise level. The units in the above are in feet or degrees. Compute the position at the start and end of the sample interval (the sample interval is 480 ms long when using 64 sample mode) return the maximum amplitude of these two values. The result is multiplied by the system transfer function.

AMPLITUDE EQUATION

The amplitude for a single target is given by:

Amplitude(R,FREQ,THETA,G,BW,Gamma)=G*FCT(FREQ)*EXP(BW*THETA^2)/R^Gamma

R is the target to radar distance in feet

Rdot is the time rate of change of R

THETA is the angle that that target makes with the centerline of the beam.

G is a constant. It has a nominal value. The beam to beam variation is 10% of nominal.

FCT(Freq) is the response function (see table)

EXP( ) is the exponential function

BW is a beam width parameter

Gamma is expected to be 2.00, but appears to be about 1.5 +- 0.2 based on MITRE tests.

MULTIPATH

The major source of noise is multipath. The multipath effect depends on the environment. Values of up to 100% have been observed. Each beam may have different amounts of multipath. As a starting point the multipath can be set to 20%.

INITIAL SETUP

The model has an initial setup function. The set of gains computed. This simulates the beam to beam differences. For each beam (3 in each radar) the gain (G) is:

G= Gain+RANDOM(-a,.a)*Gain

Where RANDOM(-a,a) generates a random number uniformly in (-a,a). The value of a is presently estimated to be 0.05. Gain is the nominal value. A value of G is produced for each beam.

AMPLUTUDE AND FREQUENCY GENERATION—Single target

The amplitude (AMP) and frequency (FREQ) measurements for a single target is generated as follows:

Compute R1 and THETA1 at the start (t) of the measurement time window for the desired radar unit and beam.

Compute R2 and THETA2 at the end (t+DT) of the window.

N=INTEGER(ABS(2*(R2-R1)/WL))

FREQX=N/DT

R=MINIMUM(R1,R2)

A=amplitude(R,FREQX,THETA,G,BW,Gamma)

AMP=A+A*RANDOM(-0.2,0.2)

If AMP < AMPTH set FREQ=0

If AMP >= AMPTH set FREQ=FREQX

The above computes the measurements for a single target. Note that each beam has different values for G, BW and Gamma.

MULTIPLE TARGETS

The radar does not resolve multiple targets. It provides a single value of AMP,FREQ for all targets. The values of AMP and FREQ are computed as follows:

Use the single target method to compute AMPi and FREQi for all targets i=1,N

Generate Ri=RANDOM(0,6.28) for i=1,N

AMPRE=Sum(AMPi*cos(Ri)), AMPIM=Sum(AMPi*sin(Ri)) , i=1,N

AMP=sqrt(AMPRE^2 +AMPIM^2)

Select the largest amplitude AMPj

Select the next largest amplitude AMPk

If AMPj >= 5*AMPk then FREQ=FREQj else FREQ=RANDOM(0,FREQj)

CODE FOR RADSIM

The above model is represented by the following code:

```
$Gain$ =  3000

$WaveLength$ = 0.105 // feet

$AmpThresholdFactor$ = 0.25 // Changed from absolute value of 40

$BW$ = 4.88e-4

initialize() {
```

```
  For each sensor SENSOR
     For each head HEAD
          G[SENSOR.HEAD] = exp(ln($Gain$) * (1 + 0.15 * NormalRandom())
          Gamma[SENSOR.HEAD] = Equation.BetaRand()
          NoiseFloor[SENSOR.HEAD] = max(5, 25 + 5 * NormalRandom())
     endfor
  endfor
}

begin_scanning_event() {
  start_time = current_simtime;
  for each moving target TARG
    Range1[TARG] = distance(SENSOR, TARG)
    Theta1[TARG] = angle(SENSOR, TARG)
  endfor
}

end_scanning_event() {
   end_time = current_simtime;
   for each moving target TARG
     Range2[TARG] = distance(SENSOR, TARG)
     Theta2[TARG] = angle(SENSOR, TARG)

     Phase = Random(0, 2*PI)
     AmpVal[TARG],FreqVal[TARG] = compute_Amp&Freq_measurement(TARG)
     AmpSum_Real = AmpSum_Real + (AmpVal[TARG] * Sin(phase))
     AmpSum_Img = AmpSum_Img + (AmpVal[TARG] * Cos(phase))
   endfor

   Interfear_Real = NoiseFloor[SENSOR.HEAD] + 3 * NormalRandom();
   Interfear_Imag = 3 * NormalRandom();
```

```
   FinalAmp = Sqrt[(AmpSum_Real + Interfear_Real)^2 +
                    (AmpSum_Real + Interfear_Imag)^2 ]

      MaxAmp, MaxTarg = select-max(AmpVal[])
   For each moving target TARG
        if (AmpVal[TARG]>($AmpThresholdFactor$* NoiseFloor[SENSOR.HEAD]))
          then add TARG to StrongEnough[]
   endfor

   for each target TARG in StrongEnough[]
        if ( (AmpVal[TARG] / MaxAmp) > Random() )
          then add TARG to Aligned[]
   endfor

   if (Aligned[] != {} )
    FinalFreq = target from Aligned[] with max(FreqVal)
   else
    FinalFreq = 0

   Return ( FinalAmp, FinalFreq )
}


compute_Amp&Freq_measurement(TARG) {
   DT = end_time - start_time
   N = Integer( Abs( 2 * (Range2[TARG] - Range1[TARG]) / $WaveLenth$) )
   FreqX = N / DT

   R = Min(Range1[TARG], Range2[TARG])
   If (R == Range1)
    then Theta = Theta1
    else Theta = Theta2
```

```
A=Equation.Amp(R,FreqX,G[SENSOR.HEAD],Theta,$BW$,Gamma[SENSOR.HEAD]
)
   Amp = A + A * Random(-0.2, 0.2)
   if (Amp < $AmpThreshold$)
     Freq = 0
   else
        Freq = Freq
   Return( Amp, Freq )
}


Equation.amp(Range, Freq, Theta, G, BW, Gamma) {

              G * Equation.FCT(Freq) * exp (-BW * Theta^2)
   amp =  ---------------------------------------------
                  Range^Gamma

   Return(amp)
}


Equation.FCT(f) {
 Interpolate using table:
   0.1    0.0
   0.5    0.087
   1.0    0.26
   2.0    0.52
   3.0    0.96
   4.0    0.98
   5.0   1.0
   6.0   1.0
```

```
   10.0  1.0
   20.0  0.93
   30.0  0.76
   40.0  0.56
   50.0  0.43
   60.0  0.28
}

Equation.BetaRand() {
  p = rand()
  q := 1-p

  u := p^(2/3)      // may need exp(2*ln(p)/3)
  v := sqrt(q)

  A := u * (.5428835237 + .1178890080*u + 1.085767047*p) * q^2
  B := (2.911111111 - 2.230109869*v - 1.911111111*p +
1.499813126*v*p) * p^2

  return (A + B)
}
```

Investigation of the RF Link Communication Throughput

To CP software must monitor the output from the comm link receiver to detect incoming messages. As other processes load the CPU, it is possible that messages will be lost. Tests were conducted by the University of Massachusetts to determine the effect of processor load on the comm link throughput. The results showed a degradation of performance at high CPU loads. This section describes the results as well as describes the approach used to improve the throughput.

Based on the data results of the Umass Ping Pong program (as shown in Table 1-1), it was shown that the ANTs Challenge Problem (CP) RF Communication software was "resending" too many messages and that the communication throughput was too low (i.e., not enough messages exchanged per second). The UMass "busyThread" was, by design, saturating the CPU and preventing the timely response of the underlying CP threads to communication activities. The Umass Ping Pong program was exchanging messages of increasing length (2, 23, 43 ... 183) inside of a loop that increased the delay time imposed by the "busyThread". The delay applied was used as an iterator (0, 20000, 40000 ... 140000) in the "busyThread's" repeated calculations of sqrt.

In the interest of time, we ran only one delay duration – 120000. We felt that the performance numbers reported by UMass at this delay rate were representative of the poor performance displayed by the software. An initial attempt to raise the Java thread priorities within the "Native Thread" model was unsuccessful because the thread model on Linux systems apparently does not allow for Java thread priority assignment via Thread.setPriority (). The Java threads all map to kernel threads (*lwp*s) and scheduling of these *lwp*s is not representative of any Thread.setPriority() priority and seems to be controlled strictly by the OS.

	2	23	43	63	83	103	123	143	163	183
Timeouts	35%	2%	62%	72%	28%	62%	56%	12%	66%	66%

Table 1-1 UMass Original Ping Pong results

At that point, an investigation of the CP software running the Umass Ping Pong program in the "Green Threads" model yielded some interesting performance numbers if nothing else. We ran a number of configurations and tests. Table 1-2 shows the Green Thread performance of the CP/UMass software running with equal priorities. As can be seen

from the data, the number of resends (Timeouts) is high; the time taken to run through the message lengths is long and the throughput is low.

	5	25	45	65	85	105	125	145	165
Timeouts	21%	40%	38%	38%	36%	37%	36%	36%	34%
Elapsed	449	508	510	518	503	521	538	538	504
Avg. msgs/sec	1.75	1.17	1.21	1.2	1.27	1.21	1.2	1.2	1.3
Umass calls/sec*1000	138	124	122	122	122	121	121	121	122

Table 1-2 Green Threads All Threads Even Priority = 5

Table 1-3 shows the Green Thread performance of the CP/UMass software running with raised priorities for the CP Communication Service Threads. As can be seen from the data, the number of resends (Timeouts) is low; the time taken to run through the message lengths is reduced and the throughput is improved.

	5	25	45	65	85	105	125	145	165
Timeouts	1%	0%	0%	0%	0%	0%	5%	17%	18%
Elapsed	156s	197s	96s	100s	240s	168s	248s	317s	268s
Avg. msgs/sec	6.3	5.05	10.4	9.96	3.44	5.94	3.83	2.61	3.1
Umass calls/sec*1000	135	139	102	96	138	114	125	126	114

Table 1- 3 Green Threads Core CP Threads Priority = 8

But, based on comments from the CP users, a “Green Threads” solution is not a desirable one within the ANTs community. A “Native Threads” solution is more palatable. I, like Regis, am very nervous about changes this late in the game – especially one as serious as modification of the Communication code. While it does not appear possible to affect the priority of the Java threads from within the thread software while running under “Native Threads”, it is possible to use the Unix command “renice” to affect the OS scheduling of the *lwp*s that represent the CP threads. Table 1-3 shows the Native Thread performance of the CP/UMass software running with a lowered priority for the UMass Ping Pong “busyThread”. As can be seen from the data, the number of resends (Timeouts) is very low; the time taken to run through the message lengths is further reduced, the throughput is further improved and the UMass software still gets a large share of the CPU.

	5	25	45	65	85	105	125	145	165
Timeouts	0%	1%	0%	1%	0%	1%	3%	4%	23%
Elapsed	81.6	91.06	94.12	100.1	104.2	110.16	128.6	139.5	218.9
Avg. msgs/sec	12.2	10.8	10.6	9.89	9.52	8.99	7.51	6.85	3.52
Umass calls/sec*1000	147	130	123	120	117	118	117	113	117

Table 1- 4 Native Threads Reduced Busy Thread Priority

As an aid to illustrate overall CPU utilization, we ran one small test with the "busyThread" at a higher priority than all other threads – thus allowing it to be the CPU hog it was intended to be. The average UMass calls per second (Table 1-5) are not altogether that much different than the previous test data shows. This shows that the CP Communications software does not require a significant fraction of the CPU time.

	5
Timeouts	100%
Elapsed	822
Avg. msgs/sec	0
Umass calls/sec*1000	146

Table 1-5 "busyThread" only at High Priority

In summary, we found that the CP software was not adversely affecting the available CPU for the busy thread (as measured by UMass calls per second) and we could significantly improve message throughput, under Native Threads, without adversely affecting the time available to the busy thread. The solution requires a startup script software to implement.

CP Test Setup

For the May 2002 CP tests BAE updated the CP software fix described in the previous section. In addition, the software was updated to allow each beam to have unique calibration data. Each beam can have a calibrated value for the gain factor (K), beam width (BW), and the radial dependence (gamma). To make this update, the following changes were requires:

The configuration file was extended to include these parameters (three sets for each node)

The tracker equation was modified to include the parameters as input instead of using fixed values.

This software was set up to run a six node test at MITRE. To test the setup, the BAE tracker as well as a tracker developed by the University of South Carolina was used. The configuration file as well as the tracker results are presented below:

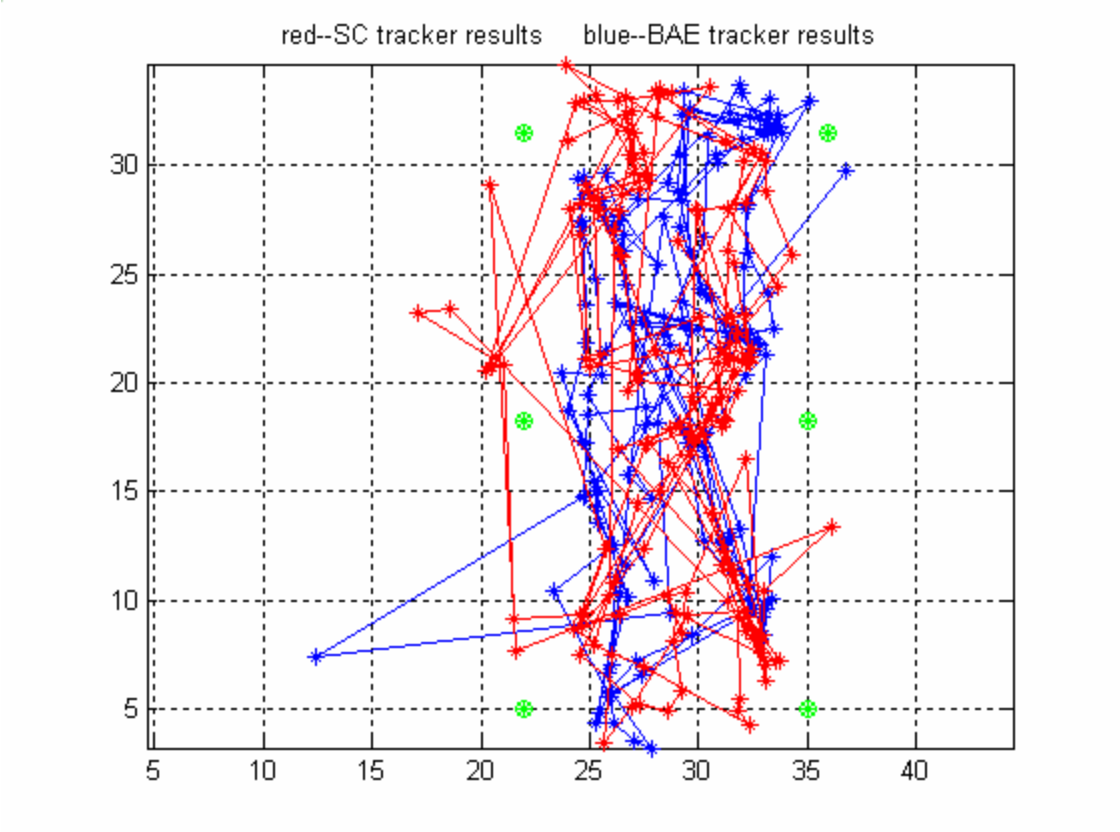
Configuration File

```
# RadSim configuration file
# Size of the simulated room
ROOMSIZE 50 40
# -- Options: Leave commented out for defaults --
# COMM_ERROR - Percent of communication error.
#   Value: from 0%-100% (default = 0, or no errors)
#COMM_ERROR 0
# MIN_TICK - Minimum real time, in milliseconds, per clock tick.
#   Default = 10 milliseconds
#MIN_TICK 10
# MAX_SIMTIME - Maximum simulated seconds to run
#   Default = infinity
#MAX_SIMTIME 60
# TARGET_START - Simulation time (in seconds) to start the targets moving
#   Default = 0
#TARGET_START 60
# NOISE - switch to turn on or off the noise
# Default : true (noise present)
#NOISE false
# LOGLEVEL - The logging level.
#   Values: 0 (no logging, defalut), 1 (basic logging), 2 (verbose logging)
#LOGLEVEL 1
# LOGFILE - Log file name.  Only used if LOGLEVEL > 0.
#   Value: a file name, default is "radsim-log.txt"
#LOGFILE radsim-log.txt
# LOGPOSITION - file name. Use by the tracker visualization tool
# Value: a file name, no default
#LOGPOSITION realdata.txt
# GRAPHICS - Show or don't show Radsim graphics.
```

```
#   Values: false - don't show graphics, true - show graphics
#GRAPHICS false
# DEBUG - Set debugging output level.
#   Values: 0 (none, default), 1 (basic), 2 (verbose), > 2 (very verbose)
#DEBUG 0
# Sensor objects:
#  SENSOR <name> <x> <y> <orientation> <K0> <Gamma0> <Bw>... <K2> <Gamma2> <Bw>
#------------------------------------------------------------------------------
SENSOR  S7      22      18.25   300     7363 2 40.0 4905 2 40.0 3000 2 40.0
SENSOR  S1      35      5       120     5019 2 40.0 2426 2 40.0 3000 2 40.0
SENSOR  S2      35      18.25   120     9914 2 40.0 3758 2 40.0 3000 2 40.0
SENSOR  S3      35      31.5    120     9319 2 40.0 4384 2 40.0 3000 2 40.0
SENSOR  S4      22      5       300     8658 2 40.0 2939 2 40.0 3000 2 40.0
SENSOR  S6      22      31.5    300     3000 2 60   3000 2 60   3000 2 60
# Mobile objects:
#  MOBOBJ <name> <x> <y> <speed> <direction>
MOB M1 10.0 10.0 0.5 0.0
#  MODOBJ <name> <x> <y> <speed> <waypoints>+
#MODOBJ M2 15.0 20.0 0.5   20.0 15.0   25.0 20.0   20.0 25.0
```

Tracker Results

The setup at MITRE used for testing has six nodes. These are shown by green markers in the figure. The configuration file lists the location and orientation of each node. Two runs were made. For this first test, calibration values obtained from the MITRE calibration program were used. These are listed in the configuration file. Note that beams 0 and 2 have measured calibration data. Beam 2 uses the default values. Additional tests using a wider beam value may improve the results. The blue curve was obtained using the BAE tracker and the red curve was from the USC tracker.



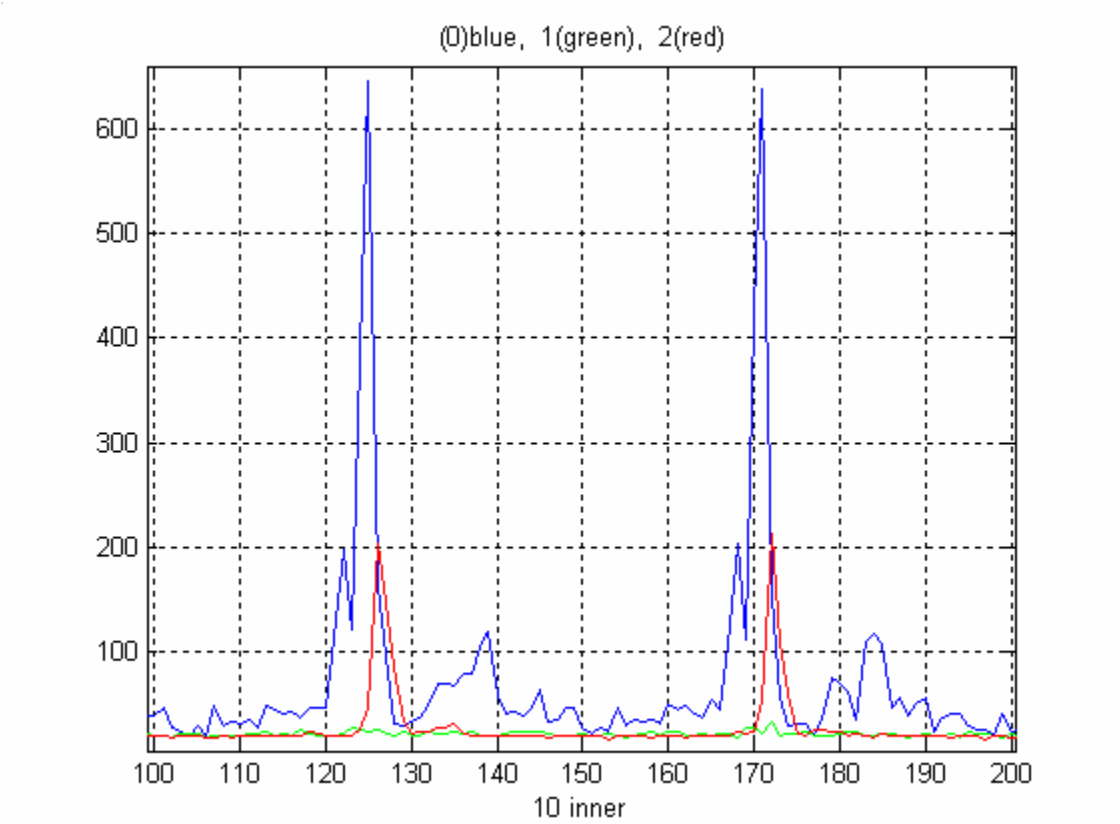
Sensor Response and Repeatability

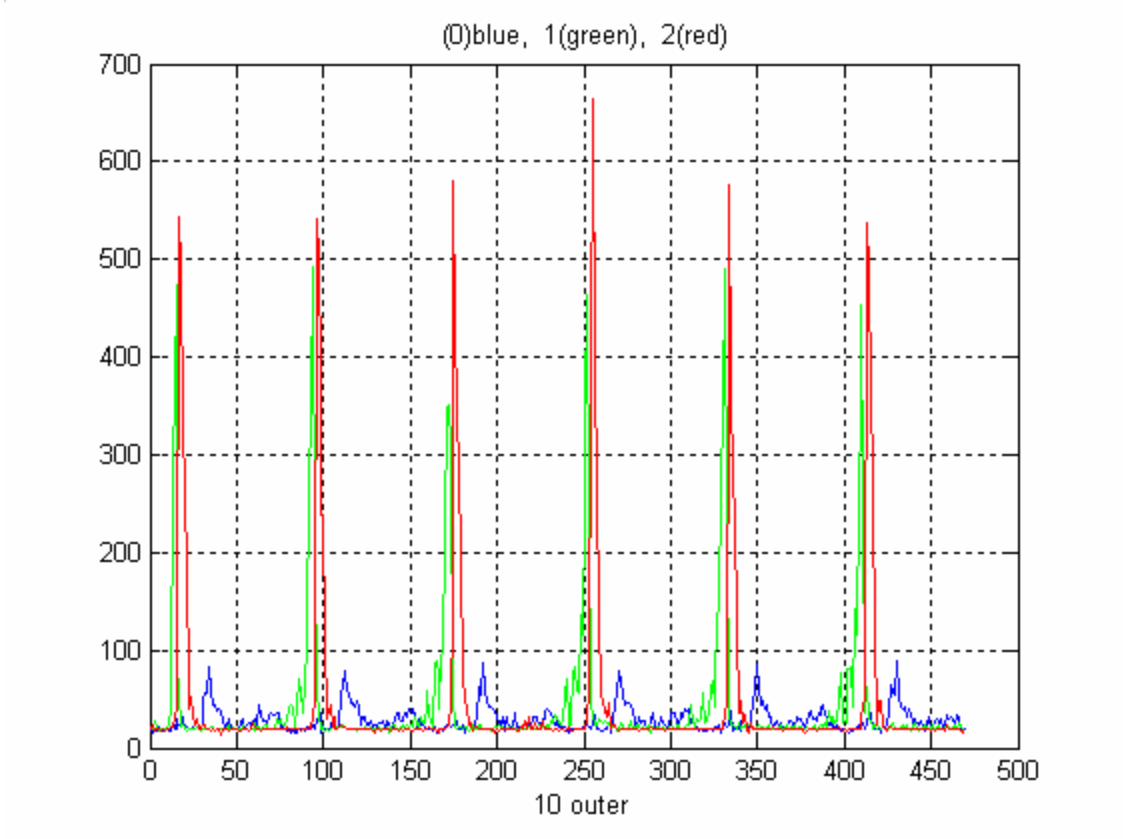
A test of the node response and repeatability was conducted at Mitre. To conduct the test, ten sensor nodes were positioned as shown below:

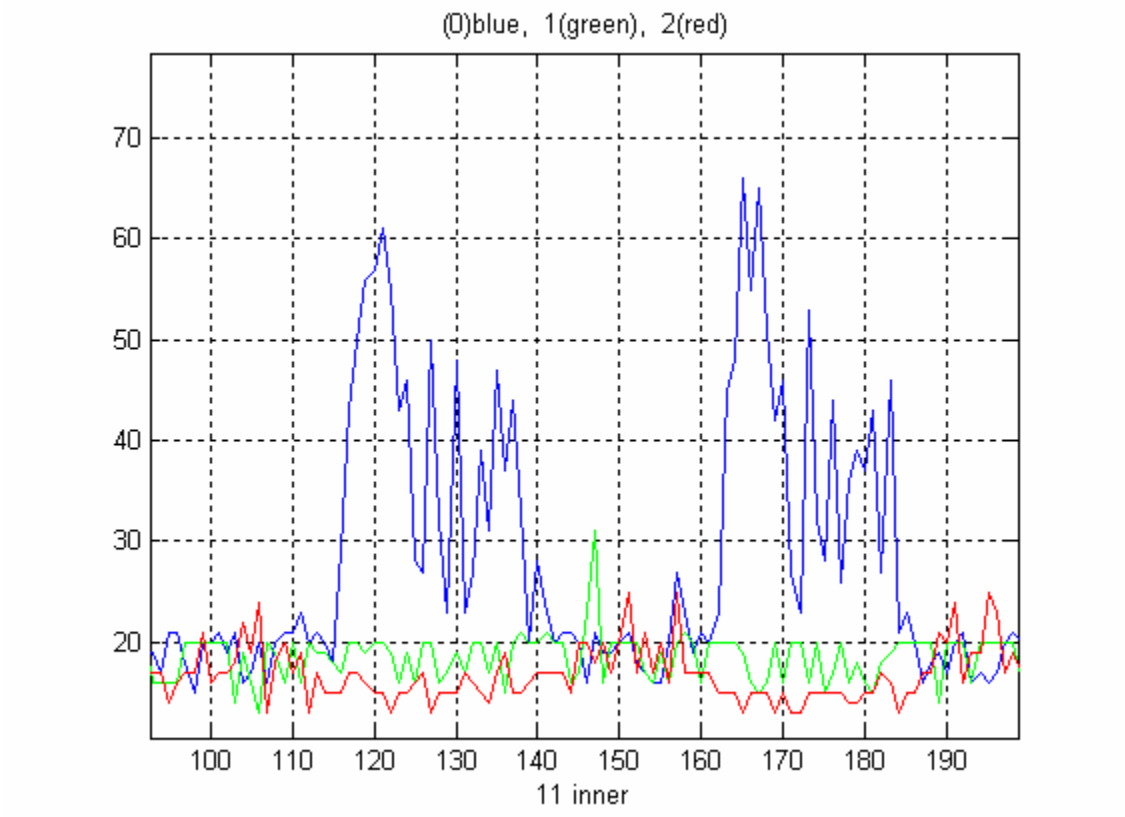
```
Sensor Locations and angles

SENSOR S13    32    14    180
SENSOR S8     27    27    270
SENSOR S4     15    27    270
SENSOR S5      7    15      0
SENSOR S6     13    10     90
SENSOR S7     14    18
SENSOR S3     26    20    315
SENSOR S9     23    11     90
SENSOR S10    32    22    180
```

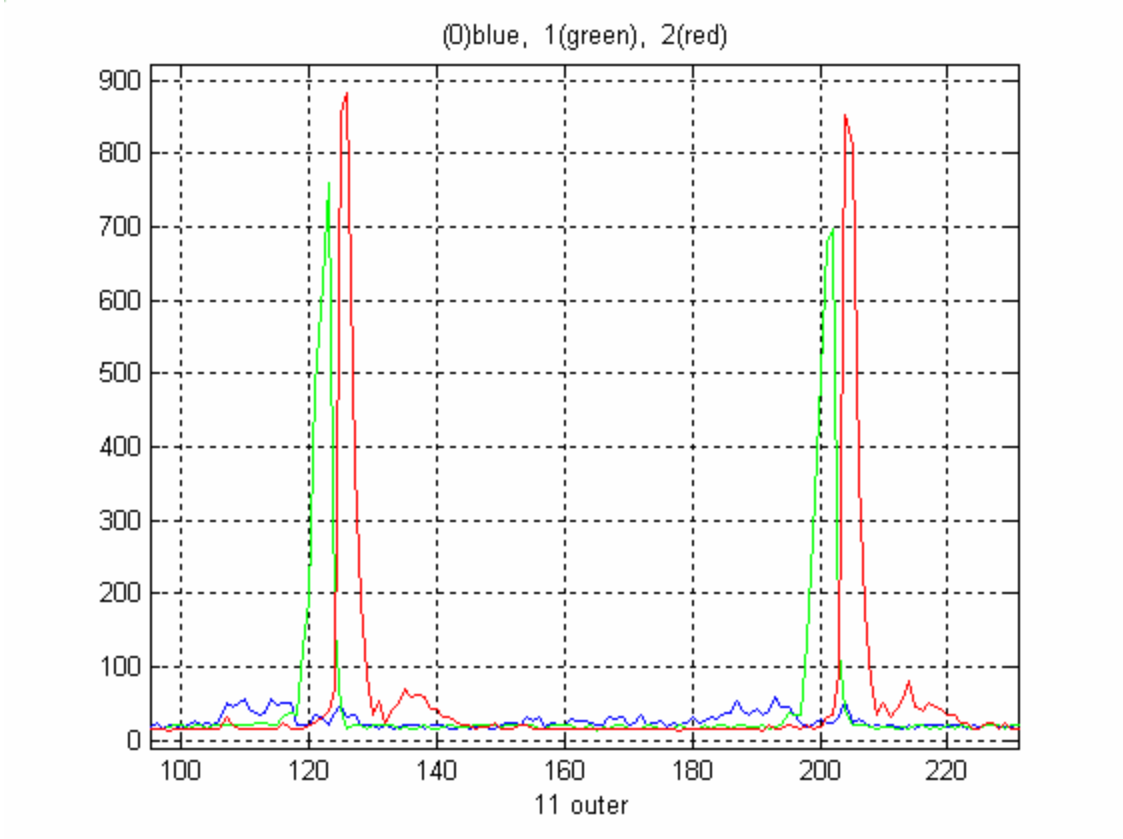
```
SENSOR S11     6    25      0
```

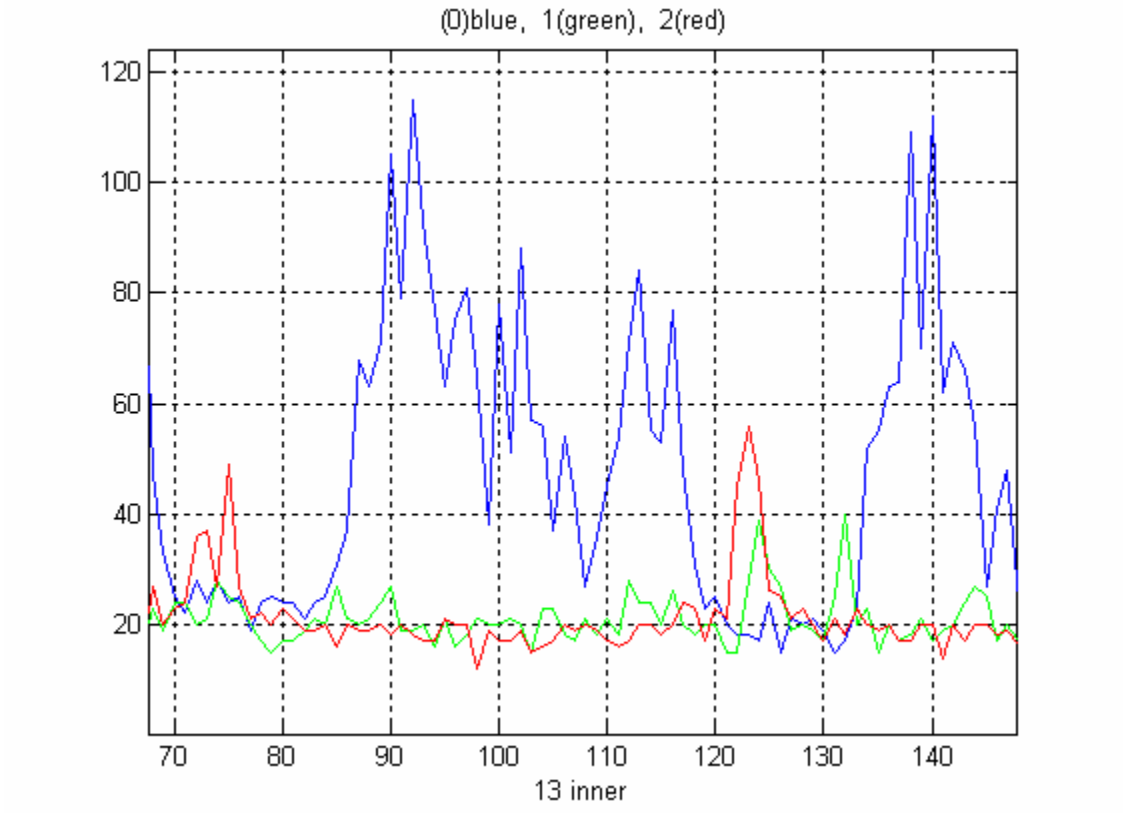
The following set of plots show the amplitude response of each radar node. The data was collected by allowing the train to loop around multiple times. A segment of the data was extracted. The segments all contained at least one complete loop around the track. The data has amplitudes from all three beams. The data repeats for each loop around the track. Note that there are 2 tracks: the outer track and inner track. Note that strong signals occur in some cases. In other cases some beams do not respond at all. This is because the beam is not aimed in a way to intercept the target.

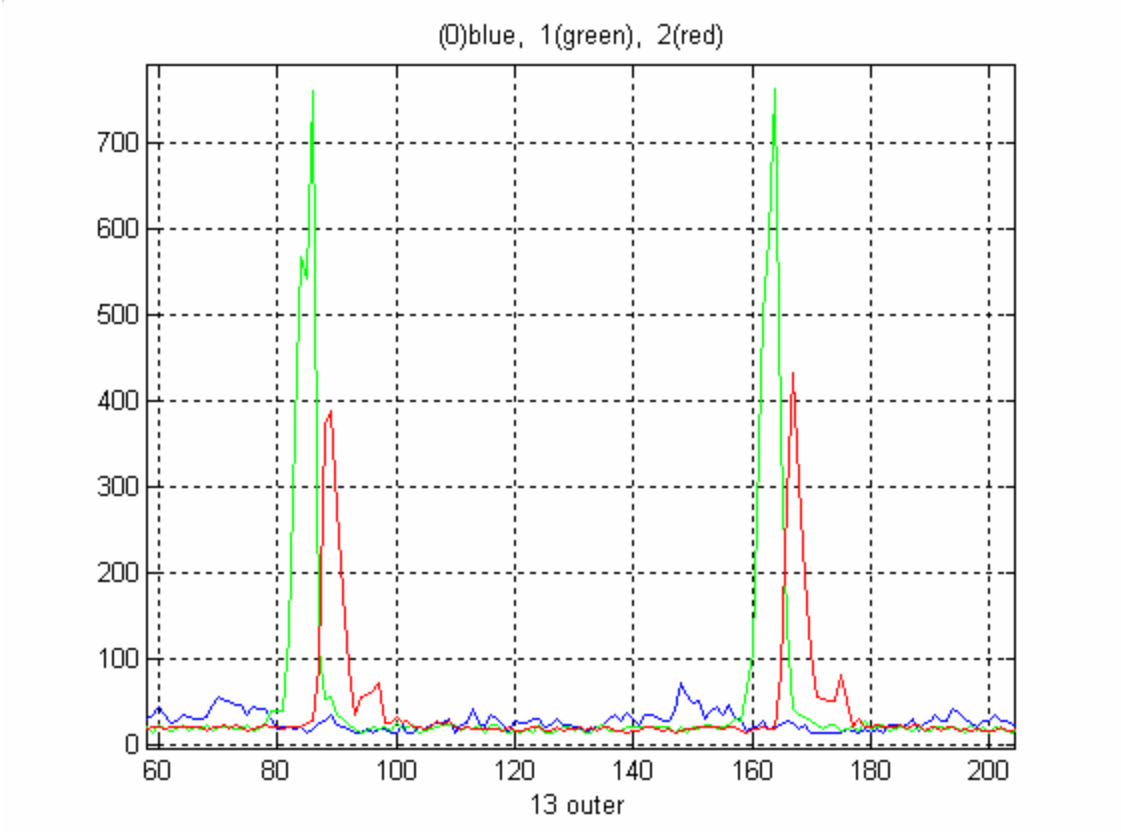


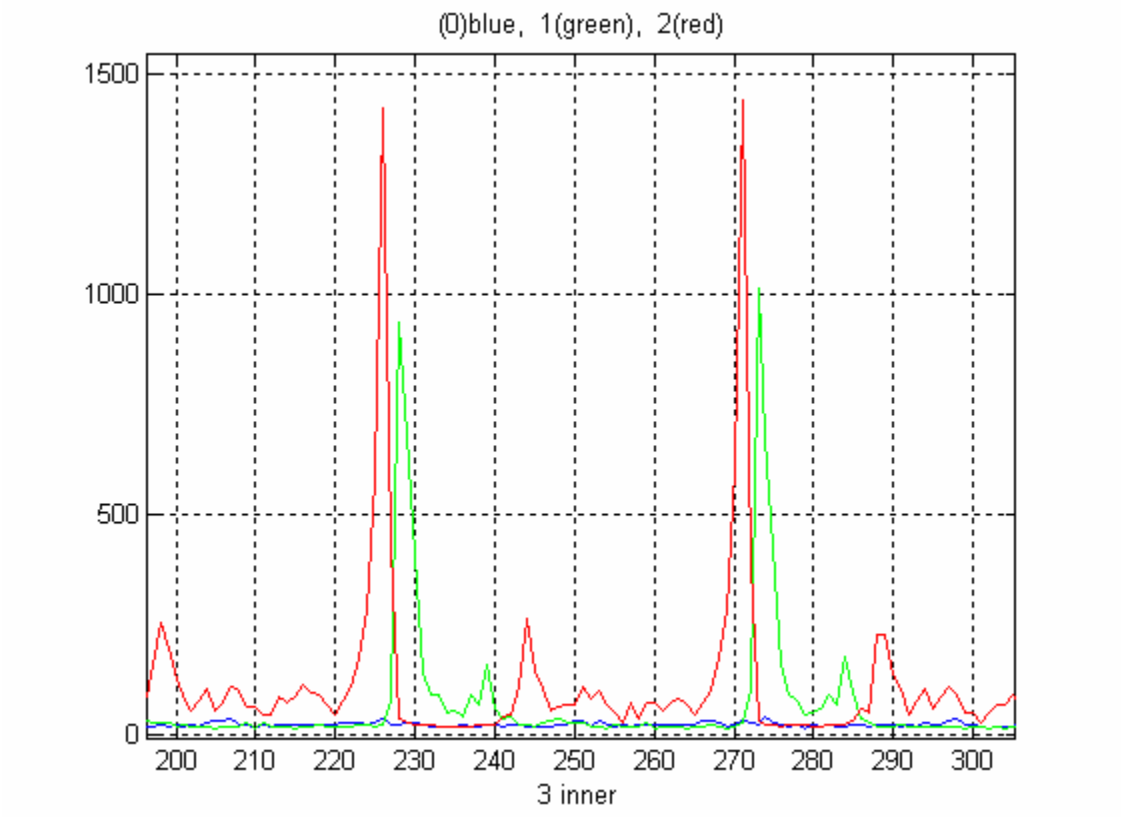

(0)blue, 1(green), 2(red)
700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500
10 outer

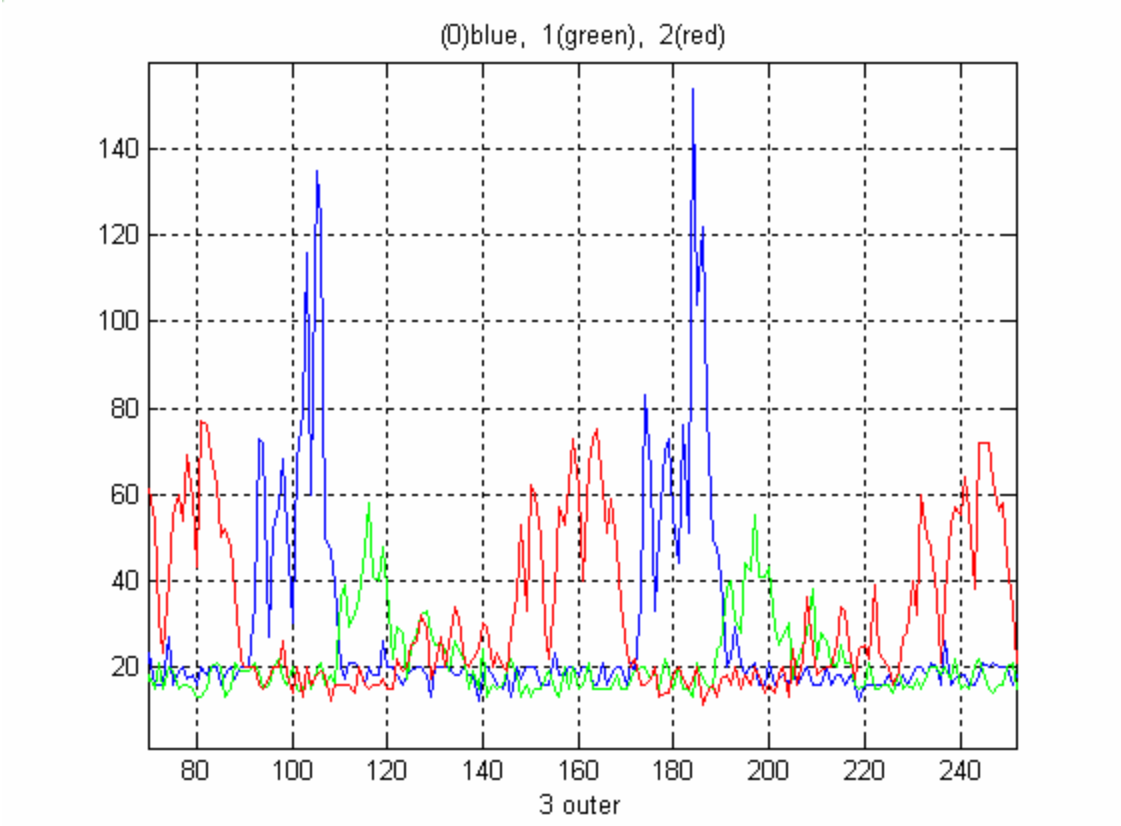

(0)blue, 1(green), 2(red)
70
60
50
40
30
20
100
110
120
130
140
150
160
170
180
190
11 inner

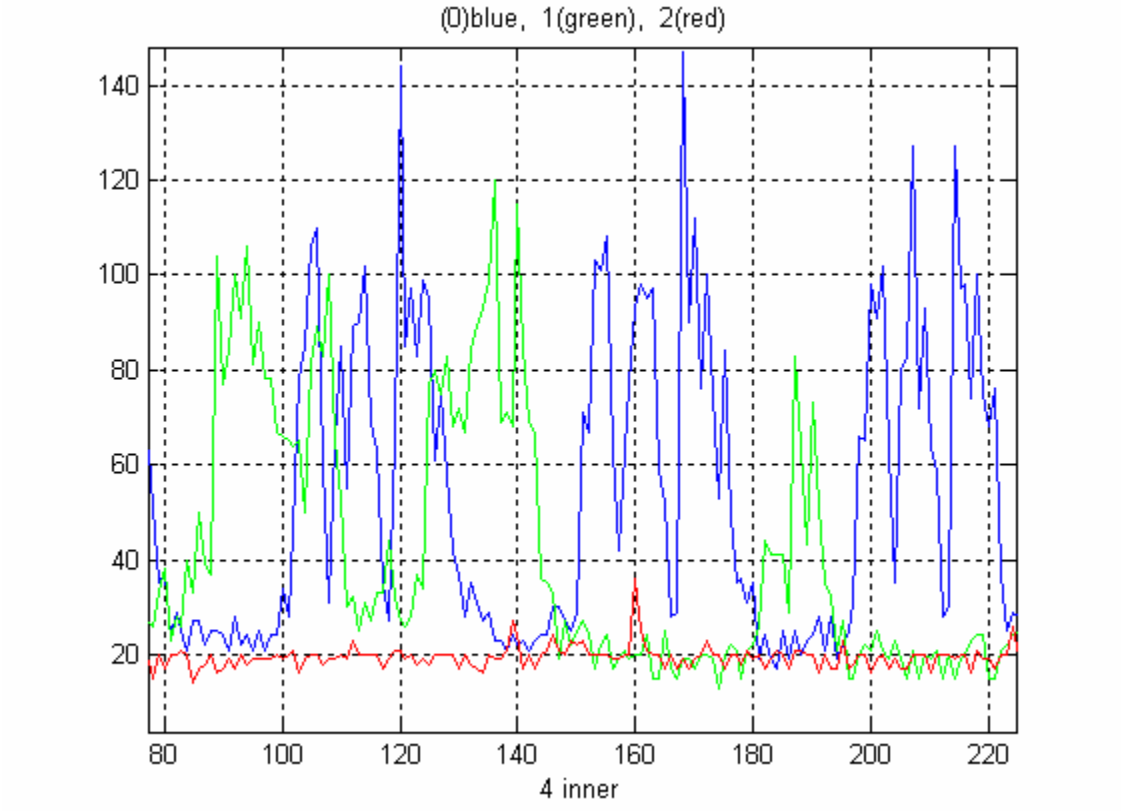

(0)blue, 1(green), 2(red)
900
800
700
600
500
400
300
200
100
0
100
120
140
160
180
200
220
11 outer

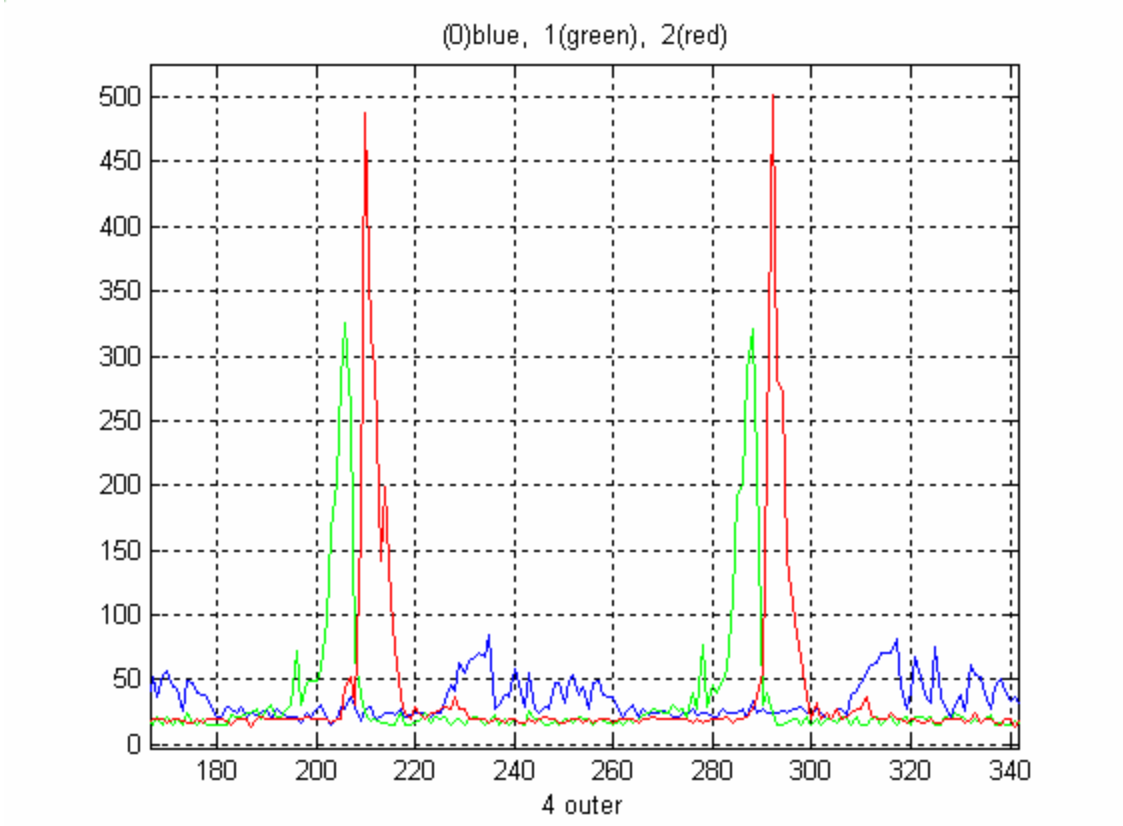

(0)blue, 1(green), 2(red)
120
100
80
60
40
20
70
80
90
100
110
120
130
140
13 inner

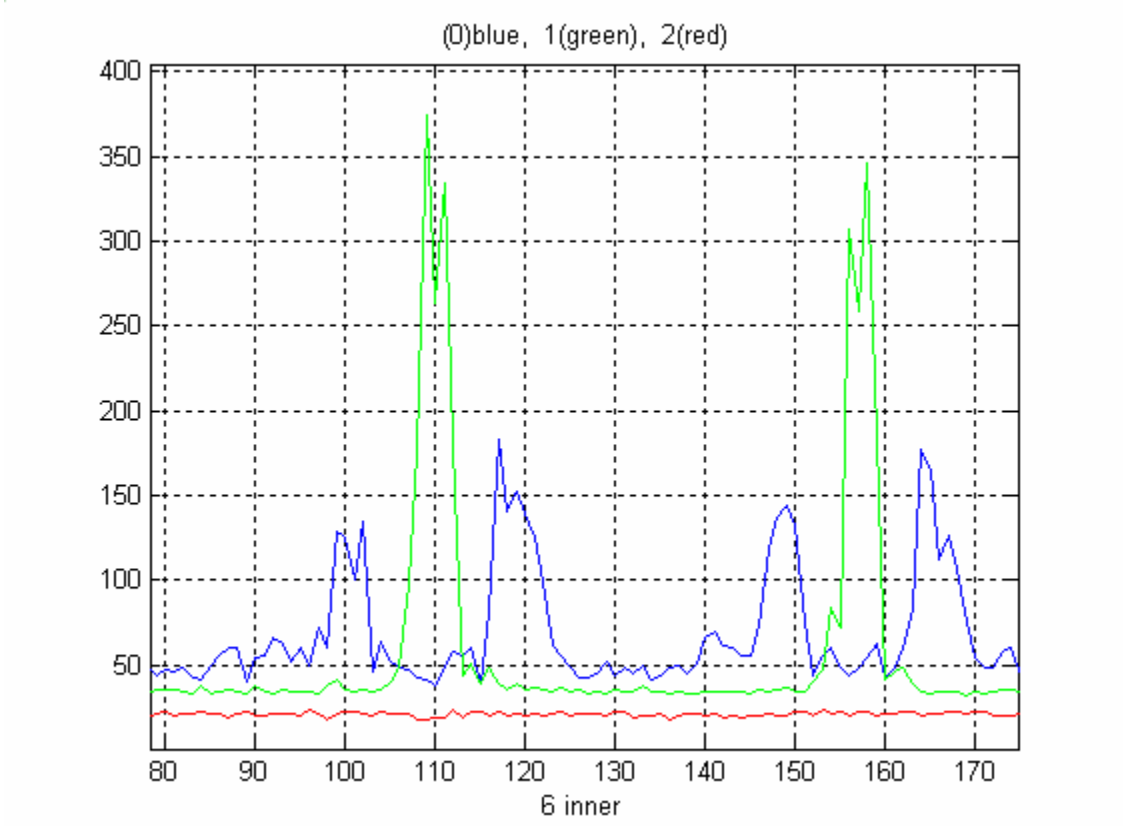

(0)blue, 1(green), 2(red)
700
600
500
400
300
200
100
0
60
80
100
120
140
160
180
200
13 outer

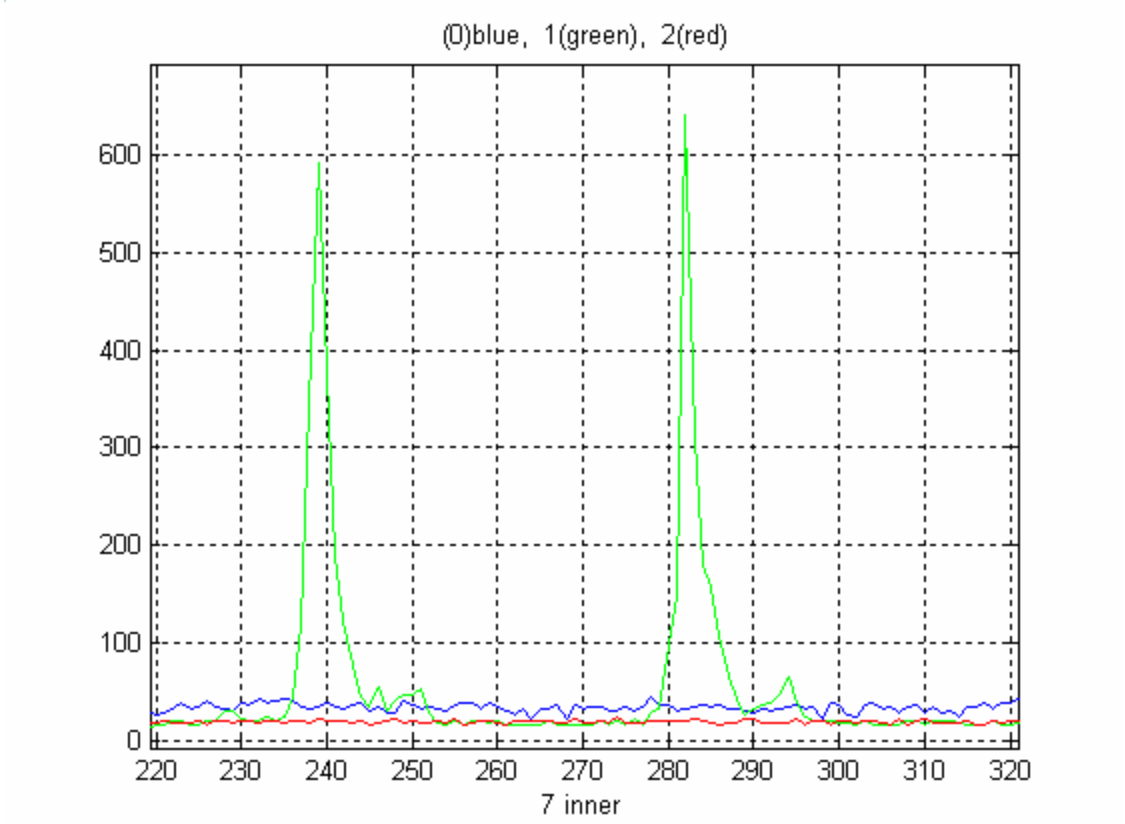

(0)blue, 1(green), 2(red)
1500
1000
500
0
200
210
220
230
240
250
260
270
280
290
300
3 inner

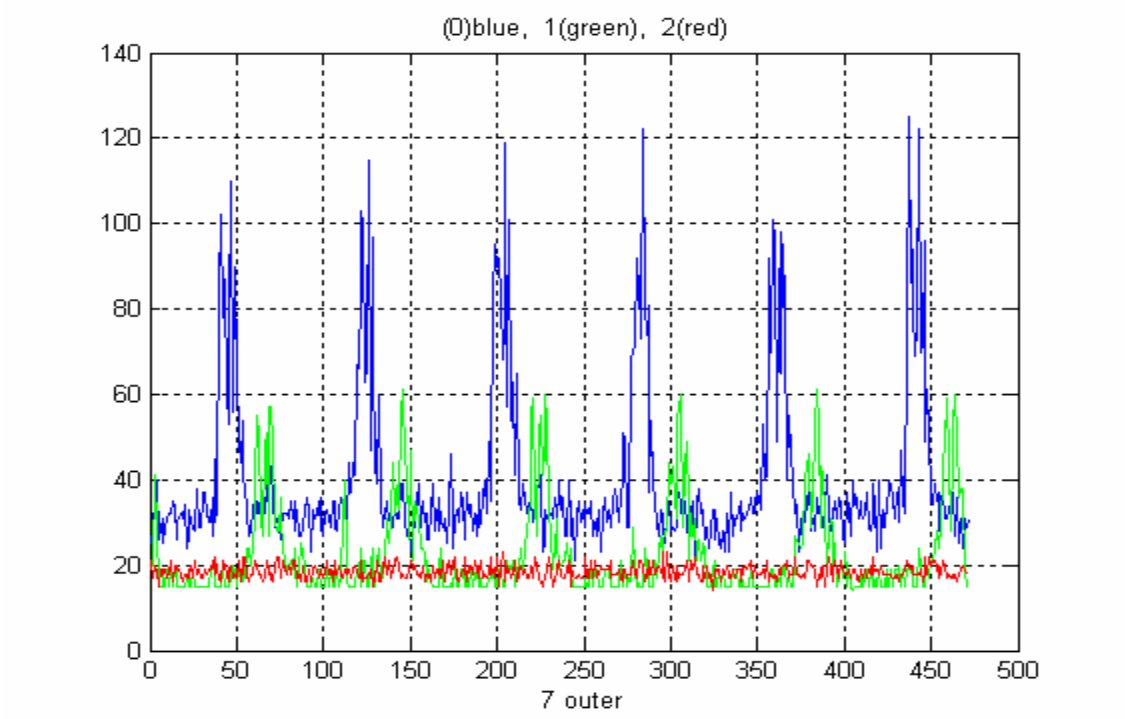

(0)blue, 1(green), 2(red)
140
120
100
80
60
40
20
80
100
120
140
160
180
200
220
240
3 outer

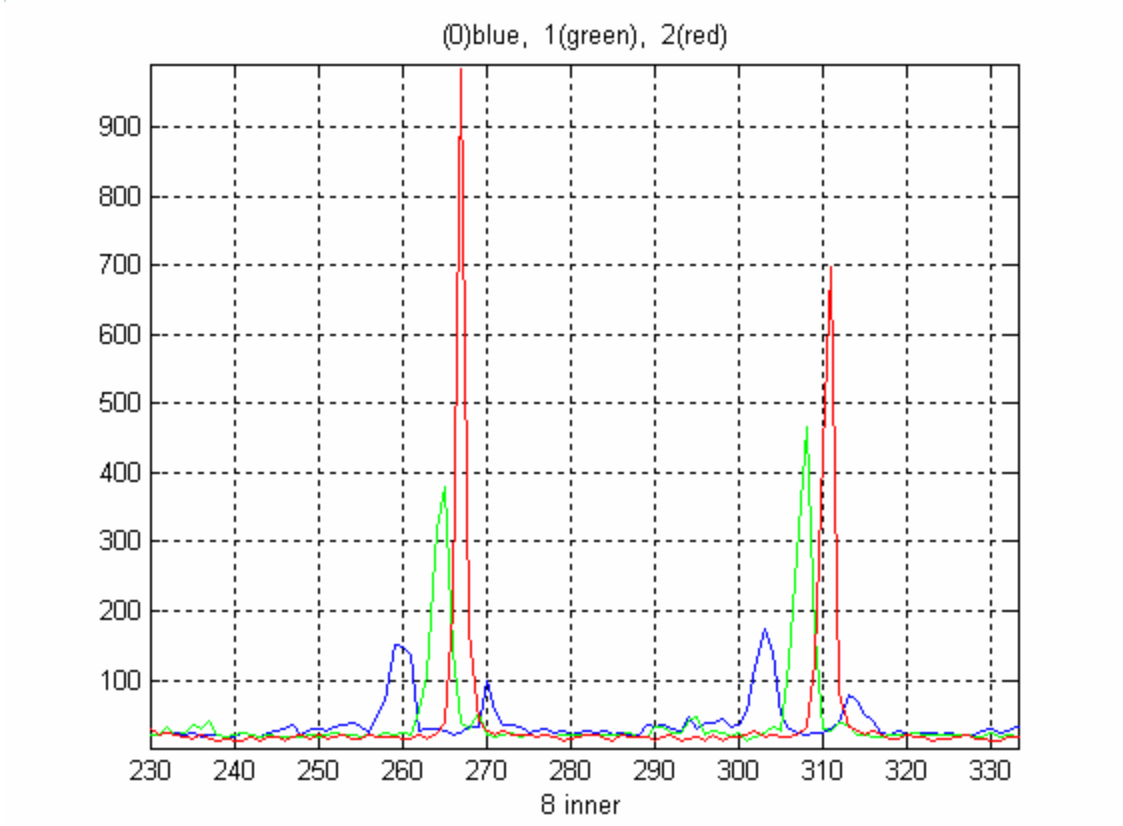

(0)blue, 1(green), 2(red)
140
120
100
80
60
40
20
80
100
120
140
160
180
200
220
4 inner

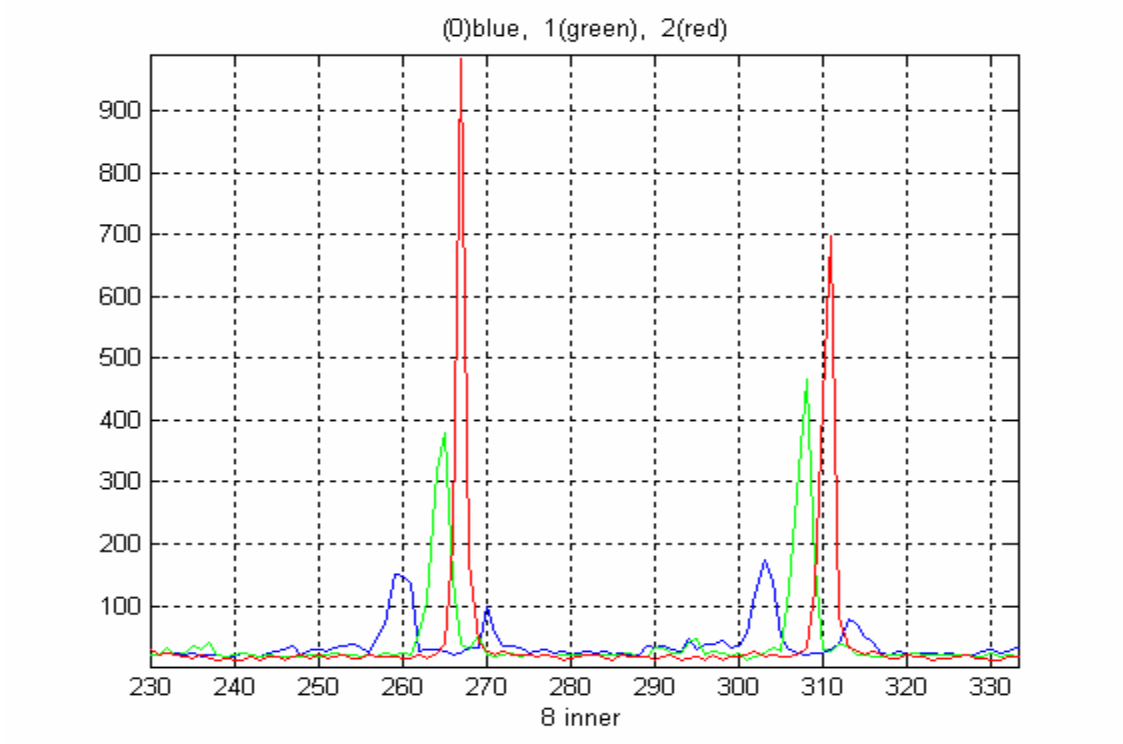

(0)blue, 1(green), 2(red)
500
450
400
350
300
250
200
150
100
50
0
180
200
220
240
260
280
300
320
340
4 outer

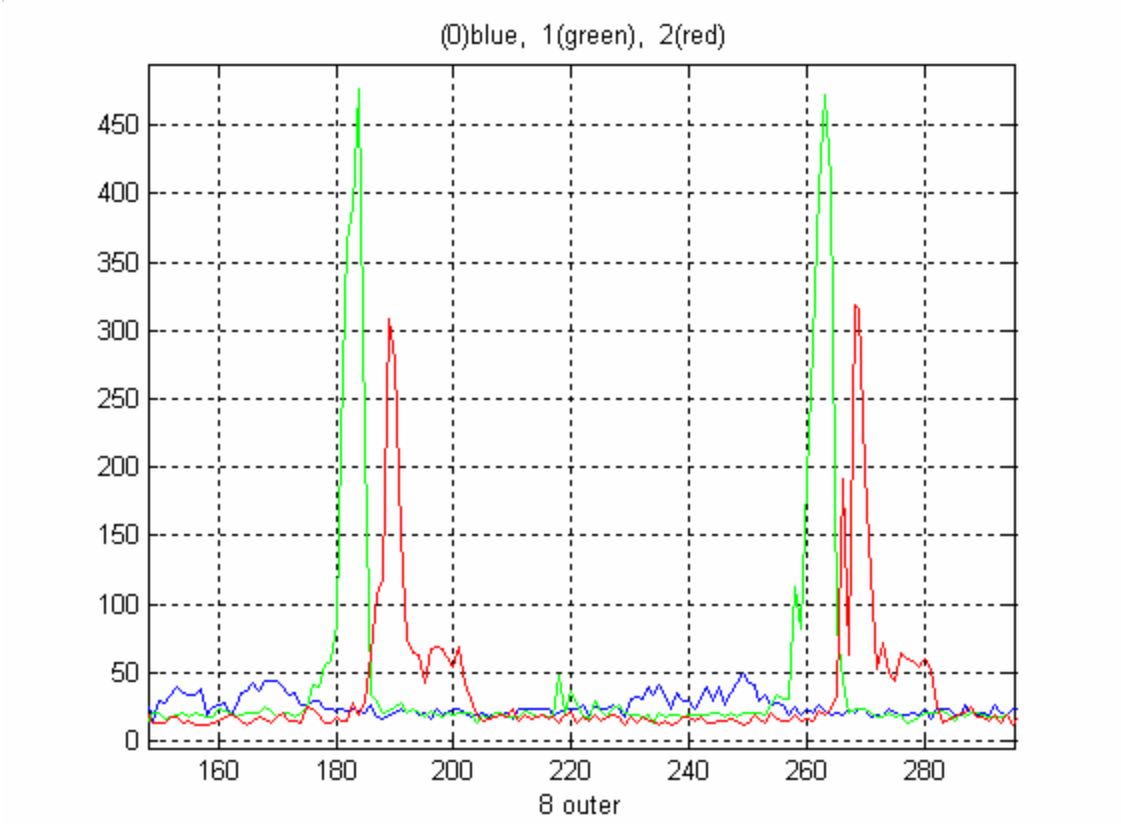

(0)blue, 1(green), 2(red)
400
350
300
250
200
150
100
50
80
90
100
110
120
130
140
150
160
170
6 inner

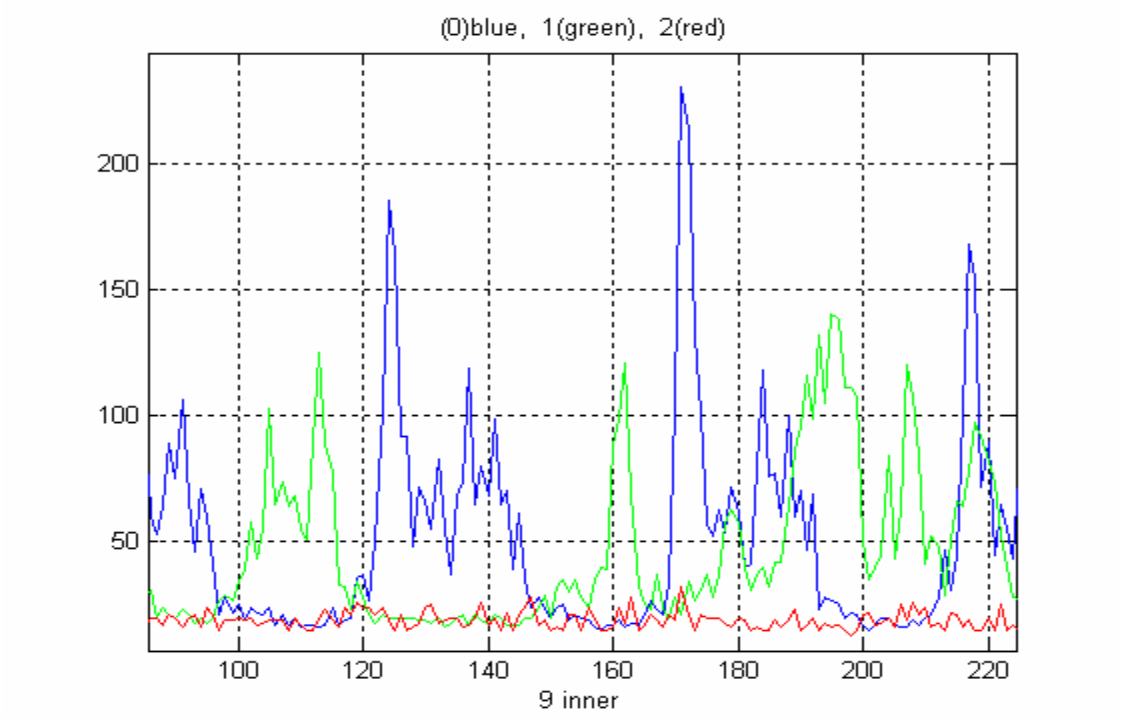

(0)blue, 1(green), 2(red)
600
500
400
300
200
100
0
220
230
240
250
260
270
280
290
300
310
320
7 inner

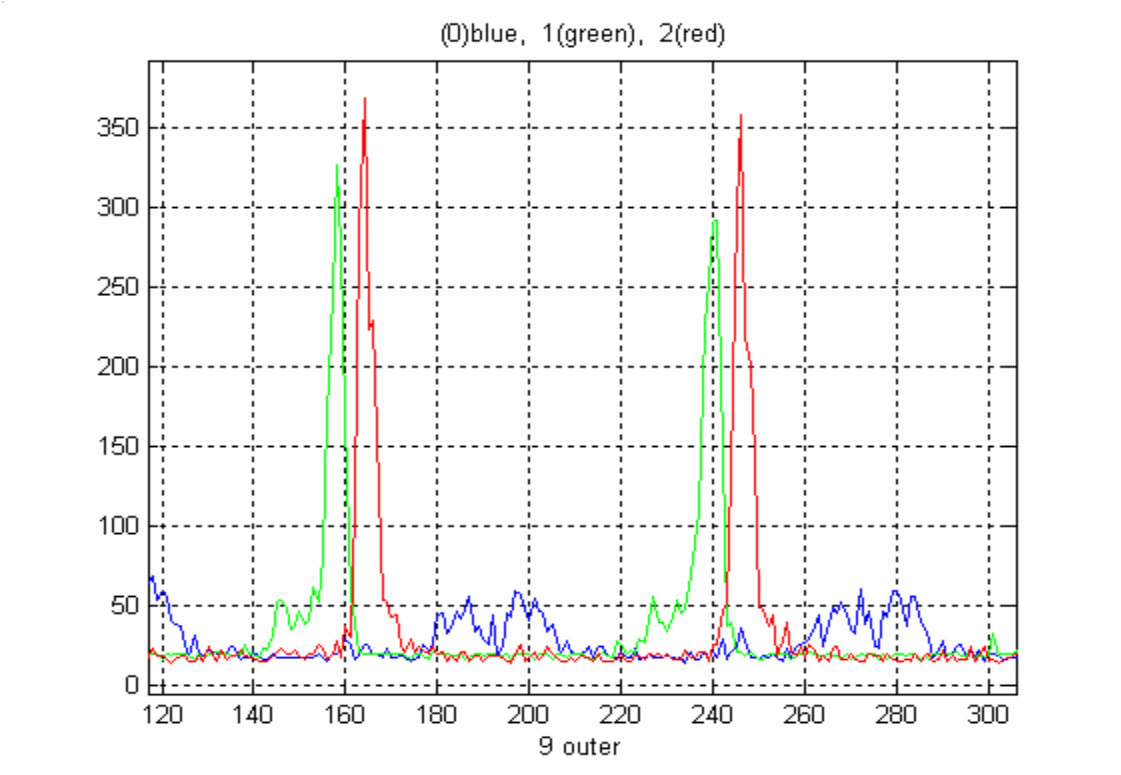

(0)blue, 1(green), 2(red)
140
120
100
80
60
40
20
0
0
50
100
150
200
250
300
350
400
450
500
7 outer


(0)blue, 1(green), 2(red)
900
800
700
600
500
400
300
200
100
230
240
250
260
270
280
290
300
310
320
330
8 inner


(0)blue, 1(green), 2(red)
900
800
700
600
500
400
300
200
100
230
240
250
260
270
280
290
300
310
320
330
8 inner


(0)blue, 1(green), 2(red)
450
400
350
300
250
200
150
100
50
0
160
180
200
220
240
260
280
8 outer


(0)blue, 1(green), 2(red)
200
150
100
50
100
120
140
160
180
200
220
9 inner


(0)blue, 1(green), 2(red)
350
300
250
200
150
100
50
0
120
140
160
180
200
220
240
260
280
300
9 outer