The Thesis committee for John Edwin Freeze

Certifies that this is the approved version of the following thesis:

Non-Myopic Sensor Management Framework for Ballistic Missile Tracking Applications

Approved By
Supervising Committee:

Maruthi Akella, Supervisor

Behcet Acikmese

Non-Myopic Sensor Management Framework for Ballistic Missile Tracking Applications

by

John Edwin Freeze, B.S.

Thesis

Presented to the Faculty of the Graduate School

of the University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin
August, 2014

Non-Myopic Sensor Management Framework for Ballistic Missile Tracking Applications

by

John Edwin Freeze, M.S.

The University of Texas at Austin, 2014

SUPERVISOR: Maruthi Akella

When hostile missile raids are launched, protecting allied assets requires that many targets be tracked simultaneously. In these raids, it is possible that the number of missiles could outnumber the sensors available to measure them. In these situations, communication between sensors can be utilized along with dynamic task planning to increase the amount of knowledge available concerning these missiles. Since any allied decisions must depend on the knowledge available from the sensors, it follows that improving the overall knowledge will improve the ability of allies to protect their assets through improved decision making.

The goal of the this research effort is to create a Sensor Resource Management (SRM) algorithm to optimize the information available during these missile raids, as well as strengthening the simulation framework required to evaluate the performance of the SRM. The SRM must be capable of near-real-time run time so that it could potentially be deployed in a real-world

system. The SRM must be capable of providing time-varying assignments to sensors, allowing more than one target to be observed by a single sensor. The SRM must predict measurements based on sensor models to assess the potential information gain by each assignment. Using these predictions, an optimal allocation of all sensors must be constructed. The initial simulation, upon which this work was built, was capable of simulating a set number of missiles launched simultaneously, providing appropriate charts to display the accuracy of knowledge on each target as well as their predicted impact locations.

Communication delays are implemented within the simulation, and sensor models are refined. In refining the sensor models, they are given geometric limitations such as range and viewing angles. Additionally, simulated measurements incorporate geometric considerations to provide more realistic values. The SRM is also improved to account for the details added to the simulation. These improvements include creating assignment schedules and allowing a time-varying numbers of targets. The resulting simulation and SRM are presented, and potential future work is discussed.

Contents

1 Introduction

In hostile missile attack scenarios, ensuring the safety of friendly assets requires that many enemy targets be tracked simultaneously. The state of the theater rapidly changes in these scenarios, introducing uncertainty. Any decisions made by military leaders or automated defensive networks depend on this uncertain knowledge, and as such the effectiveness of these decisions is highly dependent on the quality of the information available.

To reduce the uncertainty, the targets must be tracked to estimate the current position, final impact location, and time until impact. Targets must also be discriminated to ensure that limited resources are not squandered by unnecessarily tracking non-threatening objects. Objects expected to impact nearby allied assets can therefore be prioritized to ensure their protection. Targets expected to impact at these assets must be eliminated through interception. The interception process requires high precision knowledge of target position. The ability of the existing defense mechanisms to protect key assets is then directly tied to the ability of all sensing equipment to reduce uncertainty in Situational Awareness (SA) of the theater.

1.1 Problem Definition

To protect allied assets from missile attacks, an assortment of systems are available to military officials. Multi-mode radar and electro-optical and infrared (EO/IR) sensors are utilized to locate the missiles. Some of these sensors are deployed permanently to fixed locations, while others may be mounted to sea-based platforms, aerial platforms, or even space-based plat-

forms. These sensors are tasked with target search and detection, target acquisition, target discrimination, and target tracking. When required, SAM launchers designed for missile interception may also be employed.

Missile-based attacks on allied assets can come in a great variety of cases. For cases with only a few missiles launched, existing protocols can successfully protect defended assets. But in cases where these attacks involve many missiles, the number of missiles may overwhelm the capabilities of the existing systems if all sensors work independently. If communication and planning between the available sensor resources could be utilized in these situations, the tasks could be divided between them such that the required demands could be more easily met.

To utilize this communication in planning task assignment, algorithms must be implemented to allocate each sensor to the most favorable task. Since the number of targets may overwhelm the number of sensors, the allocations must not be permanently assigned. Instead, the situation must continuously be evaluated and sensor allocations must continually change. The allocation would also have to incorporate sensor capabilities, ensuring that the target is within view of the allocated sensor. It is critical that these algorithms be efficient, as dynamic task planning must occur in real-time.

This algorithm should be able to utilize communication with all available sensors to spread the necessary tasks between them. The allocations should be made considering all possible assignments to each sensor, including sensor-fusion tasks which involve multiple sensors observing the same target. This communication should consider the use of sensor fusion, when multiple measurements from differing sensors is compiled into a single, higher-fidelity

measurement.

1.2 Related Work

Extensive work has been conducted on the subject of tracking of multiple targets with multiple sensors. The most basic form of these tracking algorithms are myopic linear assignment algorithms [7],[10]. These algorithms are myopic, or short-sighted, because they determine the optimal assignment only for the next step, and must be repeated at each step. Linear assignment algorithms require that each available resource be assigned to a single target. They also assume that multiple sensor assignments to the same target will provide a summation of the effects from each individual assignment. Since this 'linear addition' property does not reflect true sensing behavior, other algorithms have been developed which account for sensor fusion capabilities [4]. These algorithms provide more realism at the cost of exponentially-increasing calculations required, as the cost must now be calculated separately for each possible combination of available sensors.

Additional improvements to myopic linear assignment algorithms include extension to non-myopic assignments [3],[6],[5]. These non-myopic algorithms create schedules of assignments for several steps, providing a sensor schedule for the given time period. Krishnamurthy investigated a Markov decision process to determine the optimal non-myopic assignments [6]. Kreucher et al. [5] utilized a cost function which allows different tasks, including identification and tracking. An advantage of non-myopic assignments is that temporarily obstructed targets can be assigned to a sensor after the obstruction has been removed, allowing sensors to measure other

unobstructed targets during steps for which the obstruction exists. This capability can also benefit situations where target trajectories are temporarily overlapping.

A significant problem with these non-myopic assignments is the extreme computational complexity. Determining the optimal non-myopic schedule can require an incredibly high number of combinations even for relatively small problems due to combinatorial explosion. This can be seen by the consequences of adding a single step to the assignment schedule; the cost for all possible assignments must be calculated based on all previous possible assignments. These computational constraints can be prohibitive to the creation of assignments in real-time. To solve this problem, some algorithms have been developed to create a suboptimal assignment schedule which can significantly reduce the number of required calculations [2],[11].

1.3 Research Contributions

The goal of this project is to develop a Sensor Resource Management (SRM) algorithm which reduces the uncertainty of the state of the theater. The SRM algorithm will have prior knowledge of sensor capabilities and locations as well as the locations of any assets which must be protected. Using this information, the algorithm must determine the optimal allocation strategy for available sensors to reduce the uncertainty in the Situational Awareness. The allocations must consider heterogeneous tasks and sensors which can have differing capabilities.

In order to test potential SRM algorithms, a simulation had to be developed. A framework for the simulation was developed previously with basic

functionality, including implementation of sensor-fusion for measurements, target tracking with an Extended Kalman Filter, and sensor allocation based on the optimization of a cost function (see Section 2). This existing simulation framework was the starting point for the research presented here, and was heavily modified over the course of the project

During this project, the primary motivation was to develop, test, and mature the previously existing methods for SRM planning. The first implication of this goal was development of the sensor allocation methods. This goal includes, but is not limited to, non-myopic sensor scheduling, assignment filtering based on feasibility, and improvement to measurement models. From the existing framework, scheduling algorithms were purely myopic, as they only considered the current situation. It was desired to implement non-myopic scheduling, which evaluates the results of taking each possible measurement at multiple times. In the existing simulation, assignments did not account for sensor limitations, even though some targets could be outside of a sensors viewing limits. Modifications had to be made to avoid assignments which are not feasible according to sensor limitations. Measurement models also had to be improved to incorporate sensor-target geometry, including range and sensor power. These improvements were intended to dramatically improve the realism of the simulation so that it would more accurately reflect the true functionality of available systems.

Other features which needed to be modeled in the test bed include communication time delays and a time-varying number of targets. The communication time delays must be modeled within the SRM algorithm to ensure that performance is achieved as desired. As the number of targets

is unlikely to be fixed for a given real-world missile attack scenario, the simulation must be capable of 'firing' new targets and 'intercepting' targets, which requires the addition and removal of targets during the simulation.

The simulation framework was the second area targeted for development. The simulation framework should support plug and play of different utilities, such as a different SRM optimization algorithm or updated measurement models. The simulation framework had to be modified to allow execution of the simulation from a planning system. This planning system would allow the user to modify parameters of the simulation, including the number of targets, when new targets are fired, sensor locations and capabilities, and locations of defended assets.

In the following sections, this thesis presents the previously existing simulation framework and SRM. Then, the modifications made to the simulation and to the SRM throughout the course of this research are discussed. Finally, potential future work and areas for improvement are described.

2 Previous Work

The entirety of the work presented in this thesis stems from a simulation created previously through cooperation between the University of Texas at Austin and Knowledge Based Systems, Incorporated (KBSI). This previously existing simulation represents the culmination of work invested prior to the start of the research outlined in this thesis. This existing product was intended to demonstrate a functioning simulation base that could be improved and adapted to test a variety of sensor management algorithms. This product was developed using MATLAB [8].

The premise of this existing simulation is that there are a number of targets flying through the air on a ballistic trajectory. While the targets are moving, a centralized command station attempts to estimate the position of each target. To do this, the command station sends an assignment to all sensors at its disposal, and each sensor returns a measurement. The goal of the simulation is to create the command station control algorithm which provides the optimal Situational Awareness for each consecutive time step. It is assumed that the initial estimates for targets are provided to the command station through a loop of the command station assignment algorithm external to the SRM algorithm, and as such the SRM is not tasked with detection of targets for this simulation. The structure of this simulation can be seen in Figure 1.

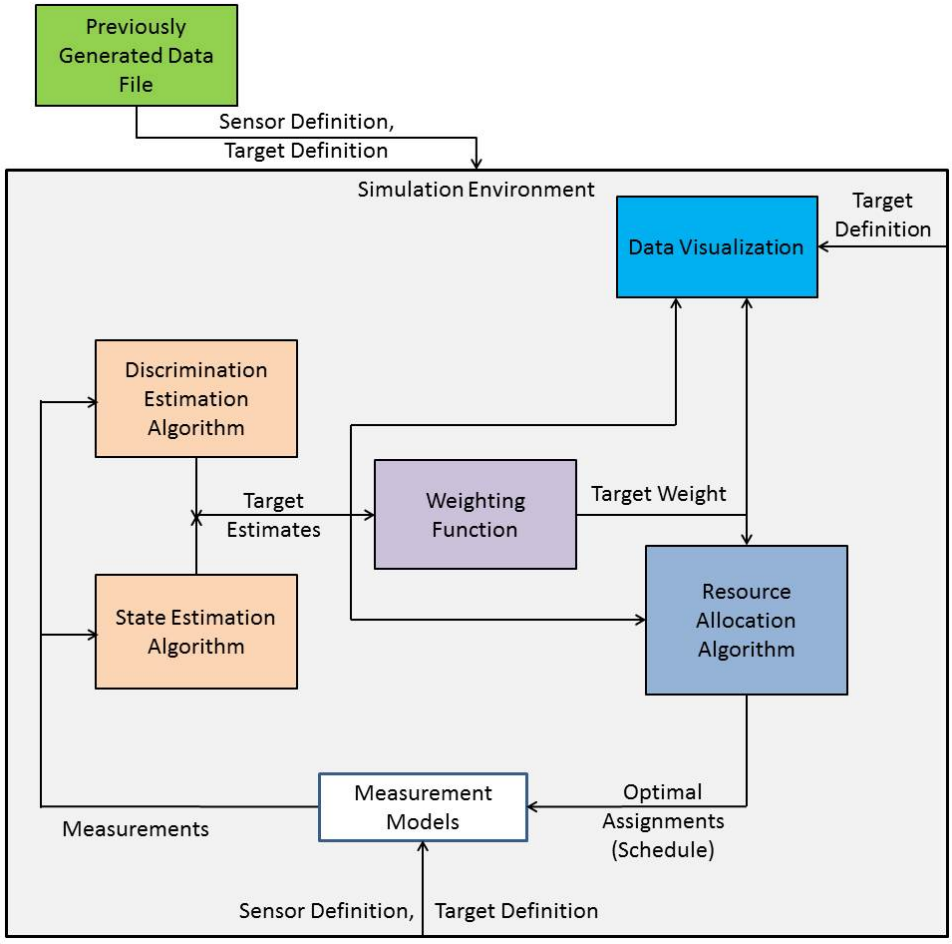


Figure 1: Representation of the Structure for the Previously Existing Simulation

2.1 Existing Simulation Capabilities

To begin, the previously created simulation used a MATLAB file to initialize the situation being simulated. This initialization file contained data on the locations of all sensors, the performance of those sensors, the location of all defended assets, and the true target trajectory throughout the simulation. The target positions were calculated previously and saved into a variable to reduce computation time during simulations.

To estimate each target's state, an Extended Kalman Filter (EKF) was used. Three dimensional position and velocity is estimated in this manner for each target. EKFs are commonly used for estimating state when receiving observations in real time, clearly matching the requirements needed for the simulation. Using the EKF to estimate target position also allowed multiple measurements from different sensors to be combined into a single estimate update with ease, as this capability is built into the EKF. The EKF also provided an estimated covariance calculation, which is a convenient tool for determining the effect a potential measurement would have on overall situational awareness.

Within the simulation, the sensors were capable of performing measurements in two different modes; one measurement mode detected the relative position of each target from the sensor, and the other measurement mode detected whether the target was a threat. As stated previously, the sensors utilized by the command station could be heterogeneous. As a result, the measurements provided from each sensor in each measurement mode could provide varying levels of detail.

Mode 1 measurements, or state measurements, may contain different information depending on the sensor. While one sensor may be able to measure the relative range, azimuth angle, and elevation angle from the sensor to the target, another sensor may only be capable of measuring these angles. Additionally, each sensor is defined with a standard deviation for its measurements. In the existing simulation, this standard deviation was constant for each sensor, and all returned measurements were normally distributed around the true target position according to the standard deviation. These measurements were used with the EKF to estimate a target's state and covariance.

Mode 2 measurements, or discrimination measurements, always contain the same information, regardless of the type of sensor. To demonstrate target discrimination, let the binary variable $X_i = \{0, 1\}$ represent the class of target i. If target i is a threat, X_i is 1. Likewise, if the target is a non-threat X_i is 0. When a sensor observes a target at time k, a binary variable $d_{i,k}$ represents the mode 2 measurement for target i. Regardless of the true class of the target, the measurement may state that the target is a threat ($d_{i,k} = 1$) or a non-threat ($d_{i,k} = 0$). Measurements of this mode could contain two types of errors in the existing simulation. The first type of error, a false positive, occurred when a non-threat was improperly identified as a threat. If a sensor had returned a 1, then the probability that the measurement was a false positive is α. The second possible error in discrimination measurements was a false negative, where a target was mistakenly classified as nonthreatening when it was a threat. If a sensor has returned a 0 for a discrimination measurement, the probability that

the measurement was a false negative is β. Each sensor was assigned a probability α and β, ranging from 0 to 1, upon initialization. In terms of α and β,

$$p(d_{i,k} = 1|X_i = 0) = \alpha$$

$$p(d_{i,k} = 0|X_i = 1) = \beta$$

For the first time step, when $k = 0$, the predicted threat level $x_{i,0}$ was initialized to 0.5, signifying no prior knowledge with respect to discrimination. Each time a new measurement was added, the predicted threat level was updated using the following equations based on the returned measurement and the prediction from the previous time step:

$$x_{i,k+1} = \begin{cases} \frac{(1-\beta)x_{i,k}}{(1-\beta)x_{i,k}+\alpha(1-x_{i,k})} & d_{i,k+1} = 1 \\ \frac{\beta x_{i,k}}{\beta x_{i,k}+(1-\alpha)(1-x_{i,k})} & d_{i,k+1} = 0 \end{cases} \tag{1}$$

If no new measurements had been received from sensors, discrimination estimates remained the same between time steps. Targets classified as threats may require different considerations in sensor allocation than those classified as non-threats, and as such threats were given higher priority in assignments.

A key feature of the command station implementation is its ability to incorporate sensor fusion. When multiple measurements from different sensors are received for the same target, the measurements are compiled into a single, more accurate measurement. Sensor fusion was implemented for observations made using both measurement modes. The command station is designed to issue assignments to sensors based on a user-selected allocation

technique. For the existing simulation, three techniques were available. The first technique, 'Sweep', consisted of measuring each target in order. Once all targets were measured, the sensors would repeat the process and measure all targets in the same order. This technique was suboptimal, but provided a good baseline with which to compare the other two assignment plans. These two optimization-based assignment plans attempted to minimize a cost function for the next time step.

For the optimization-based allocation techniques, a weighting for each target was calculated to ensure that targets deemed more dangerous were given higher priority in assignments. The weights for each target ranged from 0 to 1 based on several different estimated traits of each target. First, the distance from the target's predicted impact location to the closest defended asset was calculated. Targets with predicted impact locations nearer to the defended asset received higher weights. Second, target time until impact was also calculated, with those targets arriving sooner receiving higher weights. Target class is also factored in using the discrimination estimate mentioned previously. Through this target weighting, targets which are more dangerous were estimated to a higher level of precision than other less-dangerous targets. It is important to note that the optimization-based allocation techniques are all based on target estimates and not their true simulated values.

The first optimization-based allocation technique, 'Entropy', utilized entropy-based calculations for the cost associated with each assignment. The cost G associated with measuring the state of target i with sensor

combination j is determined by:

$$G_{i,j} = log(2\pi * |P_{i,j}|) * W_i$$

Where $|P_{i,j}|$ is the determinant of the expected 3x3 position covariance matrix of the estimate after taking a measurement of target i with sensor combination j, and W_i is a target weight ranging from zero to one. It should be noted that in order to implement sensor fusion into the cost, it is necessary to recalculate the cost function for each combination of sensors, and not merely for each sensor. These combinations include each sensor individually, all possible combinations of multiple sensors, and no sensors. As a result, the number of calculations required increases exponentially with the number of sensors available.

The second optimization-based allocation technique, 'Covariance', utilizes the estimated covariance of each target, as provided by the EKF, in the formulation of the cost function. The cost G associated with measuring the state of target i with sensor combination j is determined by:

$$G_{i,j} = (\frac{||P_{i,j,expected}||_2 - ||P_{i,j,base}||_2}{||P_{i,j,base}||_2}) * W_i$$

Where $||P_{i,j,expected}||_2$ is the expected 2-norm of the covariance after a measurement is taken of target i by sensor combination j, $||P_{i,j,base}||_2$ is the 2-norm of the covariance when no measurements are recorded. Except for the use of a different cost function, the optimization technique for the Covariance method is identical to the technique for the Entropy method.

Both of the optimization-based allocation methods utilize the same cost function with respect to discrimination assignments. This cost function is based on the entropy, and is given by:

$$G_{i,j}^{dscr} = -p(x_{i,j}) \log_2(p(x_{i,j})) - (1 - p(x_{i,j})) \log_2(1 - p(x_{i,j}))$$

Where $p(x_{i,j})$ represents the probability that the target j is identified as a threat after being measured by sensor combination i. This $x_{i,j}$ is the same as that described by Equation 1. Incremental gains in discrimination of targets were weighted higher than gains in state estimates, and as such determination of target threat level was carried out before refining target state estimates.

Once the cost function is calculated for all possible assignments, binary integer programming is utilized to determine the optimal assignment through MATLAB's Optimization Toolbox [9]. Binary integer programming is utilized because each sensor can only be assigned to a single target, and each target must be assigned some sensor combination. When the number of targets outnumbers the number of sensors, null sensor combinations will be assigned to targets and no measurement for the assignment is recorded. This optimization takes the form:

$$\begin{aligned} min(G * X) \\ subject\ to: A * X = B \end{aligned} \tag{2}$$

To implement this optimization structure, three variables are needed. The

costs G for each assignment are calculated as described above and rearranged into a row vector. The A-matrix is constructed as a matrix with dimensions $(nT * nM + nS)x(nT * nM * nC)$, where nT is the number of targets, nM is the number of measurement modes, nS is the number of sensors, and $nC = 2^{nS}$ is the number of sensor combinations. The first nS rows consist of the enumerated sensor combinations, where each column represents a sensor combination in 0's and 1's, repeated $nT * nM$ times. The next $nT * nM$ rows resemble a stretched identity matrix, with dimensions $(nT*nM)x(nT*nM*nC)$. Instead of a single '1' along the diagonal, this matrix contains nC 1's on each 'diagonal'. The rest of the entries are '0'. The B-matrix is then constructed as a column vector with $(nS + nT * nM)$ elements, all of which are 1's. The optimal X is then a column vector with $(nT * nM * nC)$ entries which satisfies the constraints that each sensor must be assigned to exactly one target and that each target must be assigned to exactly one (potentially empty) sensor combination. The MATLAB binary integer programming tool, 'bintprog', is utilized to select the optimal assignments which satisfy the constraints.

A key requirement for the simulation was that the relevant data from simulations must be displayed to the user in a concise manner while simulations were ongoing. To satisfy this requirement, the user is presented with six plots which update at each time step of the simulation (see Figure 2). The first plot shows how the discrimination parameter for several targets varies during the length of the simulation. The second plot displays three subplots. The first subplot shows the current target weights which are used in the optimization. The second and third subplots display the current en-

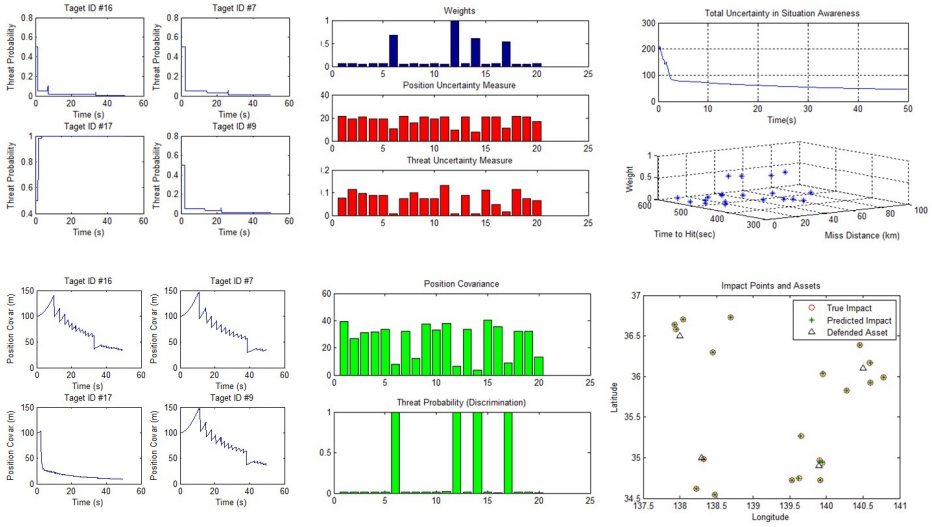


Figure 2: Sample Plots from Existing Simulation with 4 Threats in 20 Targets

tropy calculations for each of the targets' state estimates and discrimination parameters, respectively. The third plot contains two subplots, the first of which displays an overall estimate of situational awareness uncertainty. The second subplot within the third plot shows each target's time until impact, impact distance from an asset, and the resulting weight. The fourth plot, shown on the bottom left, displays the covariance of the estimates for the same targets as those shown in the first plot. The fifth plot contains two subplots, one for the current position covariance of all targets and another for the discrimination parameter for all targets. Finally, the sixth plot contains a 2-D mapping of the predicted impact locations and true impact locations for all targets, as well as the location of all defended assets. These six plots can be used to compare performance of the different allocation techniques.

2.2 Existing Simulation Shortcomings

As the existing simulation was intended to demonstrate only certain capabilities, there were many aspects that were not fully developed in the interest of providing the desired functionality with limited time. Then, aspects of the simulation which were ignored or underdeveloped could be added or improved during this research, when more time was available and more detail was desired.

The aspect of the existing simulation which required the most attention was related to sensor detail. Sensor limitations were not accounted for when recording measurements, or when performing optimization. Sensors were assumed to have infinite range and a full 360° field of view. Sensor viewing limitations could not be placed on the allocation optimization or on the recording of a measurement. Additionally, any measurement recorded by a given sensor had the same standard deviation regardless of target range from the sensor. This standard deviation was defined as constant for each sensor upon simulation initialization.

The sensor implementation also assumed that communication between the command station and each sensor was instantaneous. All assignments that were made occurred without pause, and no capability existed for recording a measurement using an assignment created in a previous time step. This inability to delay assignments degraded the applicability of the simulation to real problems, which require a delay in communication due to physical constraints. This inability to delay assignments also meant that all assignments and measurements were constrained to be myopic, or short-term, regardless

of the capabilities of the allocation optimization. In order to implement non-myopic scheduling, then, both the optimization algorithms and the sensor measurement software would need to be altered.

Additionally, all targets within the simulation were required to be known at initialization. New targets could not be identified or integrated into the estimation. Targets identified as non-threats could not be ignored, requiring additional calculations in the optimization algorithm that were unlikely to be beneficial.

3 Contributions

During this research, the primary goal was refinement of the SRM and simulation. Modifications were motivated by improving realism of the simulation. These modifications primarily included adding more details for all aspects of the simulation. Other modifications were made to improve the user interface, improving the ability to modify the simulations and to evaluate the SRM through more detailed figures.

3.1 Sensor Geometric Limitations

As mentioned previously, one of the largest shortcomings of the existing simulation was a lack of detail in the sensors. Originally the allocation algorithms and measurement models assumed that any sensor could observe a target anywhere in the air, with the only associated cost being the sensor time needed to take the observation. In reality, sensors will have a limited range within which they can make observations, and attempting to measure targets outside of this range may or may not return an observation. Even if an observation is returned, it will likely return a measurement that has a much higher standard deviation. The original model also lacked information on sensor field of view. As with range, it is was assumed that each sensor could observe any target regardless of the relative location of the target to the sensor. Many sensors, including IR cameras and phased array radars, have a specific, known field of view. Objects outside of this field of view cannot be observed reliably.

3.1.1 Adding Geometric Limitations to Sensor Models

To account for these geometric limitations, each sensor now has a defined maximum range. Additionally defined for each sensor is the boresight angle, represented by an azimuth and elevation angle, as well as a field of view, represented by a field of view angle. The boresight angle is the direction that the physical sensor is facing in the global reference frame, and provides the direction of the boresight axis. The center of the field of view coincides with the boresight axis. Targets can only be measured if the angle between the boresight axis and the target (from the point of view of the sensor) is less than the field of view angle. To improve usability of these sensors with limited field of view, some of them are built with mechanisms for adjusting their heading. Therefore, each sensor also has a defined maximum angular rate for both azimuth and elevation.

In addition to having absolute sensor limits, many radar sensors also have a variable field of view, characterized by an adjustable viewing 'pixel' for recording measurements. These viewing pixels are contained within the sensor range and field of view, and are defined by a beam angle, viewing angle, and minimum and maximum viewing ranges. The beam angle describes an axis which passes through the center of this pixel, and the viewing angle is the angle between the boundary of the viewing pixel and the beam axis. The minimum and maximum ranges provide the radial boundaries of the viewing pixel. Sensors with this capability can then record measurements for any target within the three-dimensional pixel, focusing all power available on those measurements while putting no power towards targets outside

of the viewing pixel that may otherwise be within the sensor limits. For these sensors, the field of view can be limited to a narrower cone in order to collect an observation with lower uncertainty. Alternatively, the field of view could be enlarged to assist with target acquisition, at the cost of a higher observation uncertainty. Additionally, some sensors are capable of observing more than one target if there are several present within its field of view. The cost for this capability is increased observation uncertainty based on the target's angle from the boresight axis, from the perspective of the sensor. A visual representation of these sensor properties can be seen in Figure 3.

3.1.2 Adding Geometric Limitations to the Allocation Algorithm

After modifying the sensor models to include information concerning their geometric limitations, the next step was to impose restrictions on the allocation algorithm to prohibit assignments to targets outside of sensor limits. During each time step, the geometric limitations of each sensor is compared to the estimated position of each target to identify the feasible assignments. The cost function is then calculated only for these feasible assignments. Next, a reduced form of the optimization problem from Equation 2 can be created, and a sensor assignment can be created to aim the sensor towards the desired target.

Determining which assignments are feasible occurs in two steps. First, the range from each sensor to each estimated target position is calculated. These calculations are stored in a cell matrix. Second, the azimuth and elevation angles between each sensor's boresight axis and estimated target

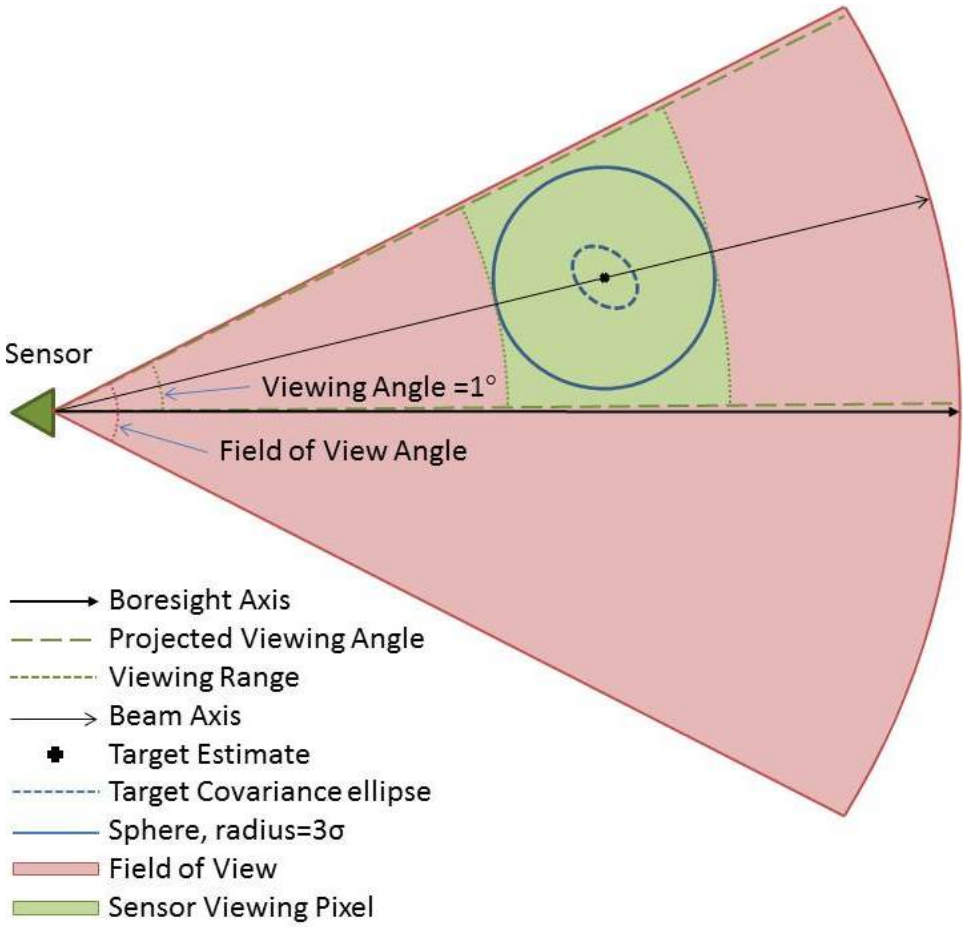


Figure 3: Two-Dimensional View of Sensor and Target Estimate Geometry

position is then calculated for any target within range of a given sensor. The azimuth and elevation angles are combined into a single angle, detailing the field of view angle required for a given sensor to observe a given target. These required viewing angles are stored within another matrix. Before performing these computations, the required viewing angle matrix is initialized as a matrix of the MATLAB variable NaN (Not a Number). For

each element corresponding to a target within sensor range, the NaN entry will be overwritten by the required viewing angle. This causes unchanged values, those target-sensor matches which are infeasible, to remain NaN. After calculating these values, a viable assignment will be one for which the required viewing angle is within the known sensor limits. It should be noted that magnitude comparisons to NaN in MATLAB return a 'false' result. Identifying feasible assignments in this way reduces computation time by eliminating unnecessary computations of the required viewing angles for targets outside of sensor viewing range.

Next, the cost is computed for all viable assignments. If any required viewing angle is beyond the sensor limits, or if any of the required limits contains NaN, then the cost computation is skipped for that assignment. Implementing the constraints in this way reduces the number of computations required for the optimization algorithm, since the cost is no longer calculated for infeasible assignments. Then, a reduced form of the optimization problem from Equation 2 can be created by eliminating the NaN elements of G. These NaN entries are the infeasible assignments, and the corresponding columns of A and rows of X must also be eliminated. It is important to note that for each infeasible sensor-target allocation, the number of entries to be eliminated from G will be equal to the number of measurement modes, as discrimination modes will be infeasible if the corresponding state measurement is. Also, if no feasible assignment exists for sensor s, entry s of the matrix B must also be changed from a 1 to a 0. It should be noted that current cost function only considers the primary target in focus, and does not measure the benefits of encompassing other targets within the

field of view.

To further improve efficiency in the optimization algorithm, consideration of the problem geometry was used to reduce the number of sensor combinations. Originally, the optimization was performed by measuring the cost function for each combination of sensors on every target. Even if two sensors were positioned such that they could never measure the same target, the cost associated with these two sensors observing each target was checked. To avoid checking sensor combinations that are infeasible, allocation algorithm sensor groups were created. These sensor groups are defined such that any sensors with overlapping ranges will be placed into the same group. Then, the optimization algorithm is executed for each sensor group. It should be noted that sensors that do not overlap may be grouped together if there is another sensor that overlaps with both sensors. Still, these grouped but non-overlapping sensor combinations will not have a cost computed due to the imposed feasibility constraints.

Once the modified allocation algorithm has provided the optimal allocations, accounting for sensor limitations, the allocation algorithm must then convert the optimal allocation from a sensor to a target into a form usable by the measurement modeling. With the implementation of sensor limitations, now the sensor beam angle is required in each sensor assignment. The sensor beam angle is the angle from the boresight axis to the beam axis, provided as one horizontal and one vertical angle. For targets that have static beam axes aligned with the boresight axes, this calculation is not necessary. But for some targets, especially radar, the beam axis can be directed to provide more accurate measurements on a desired target. The

assignment also includes the assigned mode of measurement, the intended target allocation, and the minimum and maximum range assignment for each sensor. Additionally, the resource allocation algorithm must determine the range viewing limits for any given assignment. This code utilizes the norm of the current estimate covariance. The returned minimum and maximum viewing ranges are those which minimally contain a sphere, located at the estimated target position, with radius equal to three times the norm of the estimated covariance. The sensor cone angle is configured during the simulation initialization, and remains constant throughout the simulation. For current simulations, an angle of one degree is used. While this angle is not necessarily optimal, it is representative of a typical measurement taken from a radar sensor.

3.1.3 Adding Geometric Limitations to Measurement Models

In the existing simulation, the allocation function also told the main simulation what data could be acquired on each target using each sensor. The main simulation would then use that data without checking the feasibility of taking the measurement. To mend this issue, the resource allocation function now provides the measurement models with commanded sensor azimuth and elevation angles. The simulation then checks if any targets are within the assigned viewing angle from the commanded beam axis, adding measurements if this requirement is satisfied. The process for checking feasible assignments is the same as that used in the allocation optimization algorithm, except that the true target positions are used in the measurement modeling. This structure more closely represents the true nature of the

problem, where a main control station will command a specified beam axis for a sensor, and the sensor will report what it can see at the specified heading. Any targets within the cone angle will have an associated measurement recorded, with a corresponding entry into the EKF which provides target trajectory estimations.

3.2 Radar Sensor Measurement Model

In the existing simulation, all measurements recorded by a sensor had the same standard deviation. This model is highly unrealistic, as there are a great number of factors which play into the quality of a given measurement. To resolve this issue, a radar sensor model that accounts for many of these details was created to provide more realistic measurements. This model is called each time a measurement is taken and requires the following data:

- Target range from the sensor
- Sensor power
- Sensor beam angle from boresight axis
- Sensor viewing range
- Sensor viewing angle
- Angle from beam axis to target

The further a target is from a sensor, the higher the measurement uncertainty will be. Sensor power is a static property of each radar sensor. Higher

power corresponds to a higher maximum sensing range as well as lower measurement noise. The sensor beam angle and sensor viewing limits are provided within each assignment. While a higher sensor beam angle or viewing angle will result in more measurement noise, the viewing range limits are only used to determine whether a measurement is feasible or not.

As mentioned previously, a value of one degree is used for the sensor viewing angle. In the original implementation of this radar sensor model, measurements were returned for each target within the sensor viewing angle and maximum range. As a side effect of this implementation, a problems arose. Many targets could be measured by a single sensor with no loss in fidelity. Although certain types of sensors are capable of returning different measurements for multiple targets within viewing range, it is highly unrealistic that these measurements would be of the same quality as a measurement of a single target. In reality, measurements near the center of the cone would be much more accurate than those toward the edges of the cone. Originally it was assumed that each target measured lied close enough to the beam axis that no other consideration was necessary.

To correct this problem, knowledge of the angle from the beam axis to the target was added as another input into the measurement model. This information was used to reduce measurement accuracy of targets off of the beam axis. For targets located directly on the beam axis, measurement sigma remains roughly the same. Standard deviations of measurements for targets located away from the beam axis experience an increase related to the angle difference. Additionally, measurements will only be available for targets located within the range limits provided in the assignment algorithm.

Still, whenever many targets are within view of a sensors viewing pixel, the software provides measurements for each. Since the angles required for this accommodation are already calculated to check that the target is within sensor view, this modification results in very little change in computational requirements for a large improvement in practicality of the simulation.

3.3 Communication Delay Between Sensors and the Command Station

In the existing simulation, each assignment was implemented immediately by each sensor, and each sensor immediately returned a measurement to the estimation software. A more realistic scenario includes communication delays between the command station and the sensors. Communication delays will cause a delay between when an assignment is sent to a sensor and when the sensor performs the commanded measurement. Additionally, the delay will cause the state estimation software to receive the measurements at some time later than the measurement was actually recorded. To provide more realism, the delay between the main program and each sensor can vary. As a result, it is possible for measurements that were taken at the same time to arrive at the estimation software at different time steps. To allow the measurements to arrive at different time steps, a new sensor communication function had to be adopted.

3.3.1 Adding Delays in Passing Assignments to Sensors

The first task for the sensor communication function is to store delayed assignments for a prescribed delay period. This delay period is determined

through a random number generator individually for each sensor at each time step. The average delay and standard deviation of delay for each sensor can be configured prior to simulations to improve realism. To store the delayed assignments, a 'persistent' variable is defined which retains its values across calls to the same function. Each assignment is tagged with the time it is received and the delay for each sensor.

After the incoming assignment is stored, the communication function checks for the most current assignment which has been received for each sensor after accounting for the delay period. These current assignments are then extracted from the stored assignments and passed to the measurement algorithm.

3.3.2 Adding Delays in Passing Measurements to the Command Station

Once the measurements are returned from the measurement algorithm, they must be stored until the second delay period has passed. This delay period is calculated in the same way as the first delay period, but the calculation is separate so that the delays can be unique. Once the delay period for a measurement has passed, the sensor communication function returns the tagged measurement to the command station for processing in the estimation algorithms.

The original implementation of measurements was prohibitive to saving measurements and sending them at a later time. This implementation involved sending a theoretical measurement for what each sensor recorded for each target, as well as a data type mask which described what was ac-

tually measured. If there was no measurement of a target by a sensor, the theoretical measurement was still passed along, with the data type mask signaling that the estimation should neglect those theoretical measurements. A great deal of unnecessary information was passed along in a format that made extracting a single measurement at a later time rather complex.

Now, only the recorded measurements are sent back to the estimation software. If no measurements are recorded, a null measurement matrix is sent back (see Figure 4, Step i and Step i+1). Each measurement is tagged with a corresponding sensor and target number, measurement time, the data type mask, and measurement standard deviation. The data type mask is used by the estimation software in the Kalman filter to decide what data the sensor is capable of measuring. For example, an IR sensor cannot measure range but will provide azimuth and elevation angles. The corresponding data type mask stores that information. The measurement standard deviation is also used by the EKF to determine how much to trust the new measurement.

3.3.3 Adding Delayed Measurements to the State Estimates

Once the measurements are received by the estimation software, they are used to update the estimates. When measurements have been delayed, it is possible that two measurements of the same target at the same time will arrive at different times in the estimation software. These measurements could be used to provide a more accurate estimate according to the sensor fusion model. The current estimation software can account for the change in uncertainty due to these fused measurements, but only if they are input into the estimation code at the same time. To satisfy this constraint, the

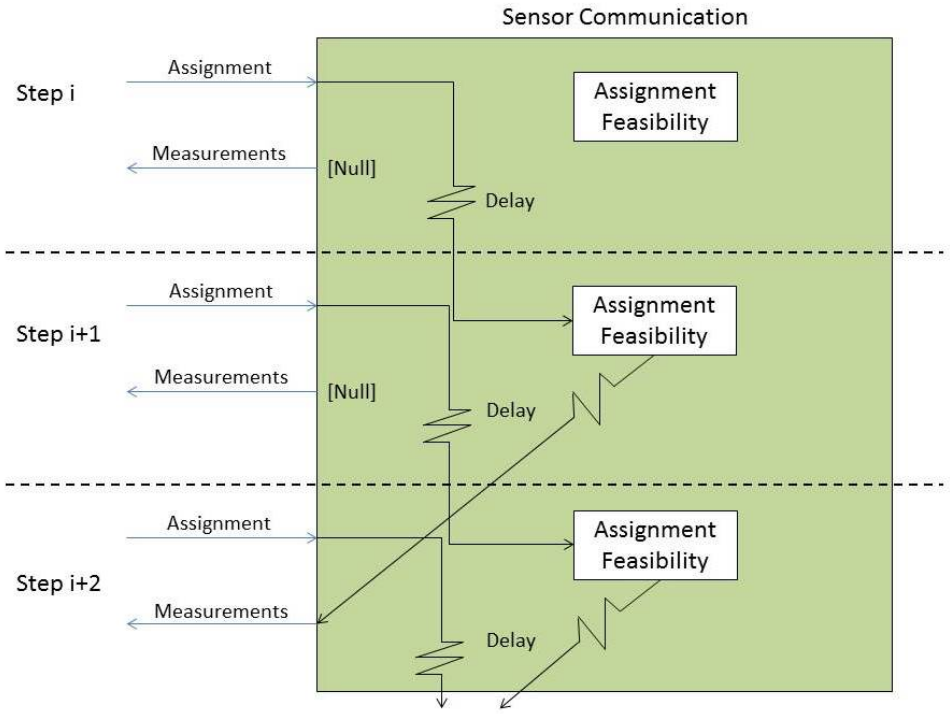


Figure 4: Sensor Communication Representation with Communication Delays

estimation code must step back to a previous time step any time a new delayed measurement is received for a target. The target state estimates are saved at each time step, and whenever new measurements are received, the estimate reverts to the time step just before the most-delayed of the new measurements was taken (see Figure 5, Step i+2). Then, the estimates at each step are updated using all available measurements until the current time. If no delayed measurements are received, no reversion is necessary and the state estimate from the previous time step is used (see Figure 4, Step i+1). Finally, the same estimate update laws are used to provide the

estimate for the next time step, and the simulation continues as before.

3.3.4 Creating Schedules of Assignments

To allow non-myopic scheduling capabilities in the future, the sensor communication software was designed with the additional capability of receiving assignment schedules. These schedules are a list of assignments intended to be carried out in sequential time steps. While a delay will cause sensors to receive the assignments after they were actually commanded, sensors following such a schedule see no further assignment delays once the schedule is received. The software allows the assignment schedules to be staggered, so that a previous assignment schedule can be carried out while waiting for an updated schedule. When the updated schedule is received, the sensors can then switch to the new schedule. With careful planning of assignment schedules, it is possible to reduce the influence of communication delay on the overall system performance. For instance, if the initial steps of the schedule are skipped, the later portion of the planned assignment schedule will occur at the times expected instead of being delayed.

To take advantage of this strategy, the sensors follow the assignment within a given schedule at the appropriate time. For instance, if a schedule is received by a sensor between the second and third time steps after it was sent, on the third time step the sensor will use the third assignment defined in the new schedule. This feature assumes that some form of time synchronization, such as GPS, is available to the components of the system.

The allocation algorithm also required updates to provide the sensor schedules. The allocation algorithm currently assumes no measurement up-

date to the target estimates between each assignment in a schedule. As a result, the position and covariance of each target is merely propagated forward at each time step according to the update law used in the EKF with no sensor measurement. By using the expected estimate of each target for the assignment, the range limitations will be based on a more realistic estimate, giving a better chance of capturing the intended target within the measurement pixel. The assignment schedules operate on the assumption that no information has been gained from any measurement in the schedule. As a result, the schedules tend to repeat assignments and are still not truly optimal schedules. To create the schedules, two methods are available. Through the first method, the optimization algorithm may be run for each time step of the schedule. Then, the allocation algorithm creates the assignment for each allocation. Alternatively, the optimization algorithm may be computed only once, and the allocation algorithm will create sensor assignments for the same allocation at each assignment in the schedule.

3.4 Multiple Discrimination Measurement Modes

In the software produced through previous development, sensors were capable of recording measurements in two modes. This measurement resulted in a discrimination parameter which would range between zero and one for each target, where a one signified that the target was a verified threat. In reality, there can be more discrimination measurement modes available to each sensor, with each mode detecting a different type of feature for the target. To reflect this, the number of modes has been increased, and can be set before initialization based on the current capabilities of available sensors.

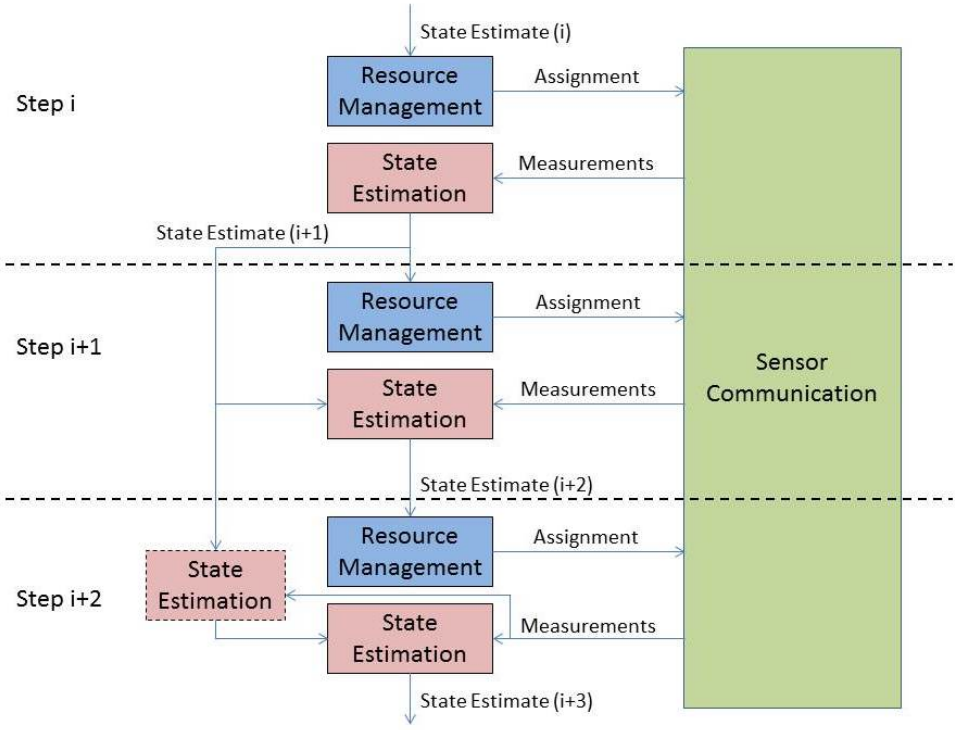


Figure 5: Updated Simulation Structure, Including Delay in Communication and Backstepping in State Estimates

To accommodate multiple discrimination modes, a new algorithm was constructed based on Bayesian Network probability models to provide a single discrimination estimate from knowledge gathered by multiple discrimination modes.

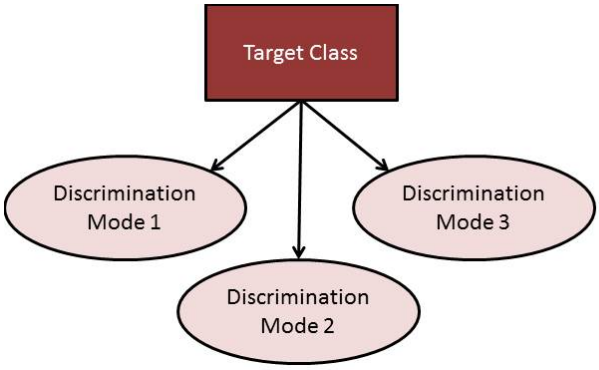
Before the Bayesian Network could be implemented, the underlying framework for the discrimination mode had to be expanded to accomodate new modes. The target class variable, which contained the discrimination estimate for each target at each time step, was the root of this expansion. This variable now contains the estimated discrimination parameter based off

of each individual discrimination mode as well as the overall discrimination estimate derived by combining the separate discrimination modes at each time step for each target. The allocation optimization algorithm was also expanded to check the cost of measurements using each discrimination mode on all targets. As a result, the G matrix and A matrix used in optimization had to be adjusted to account for these multiple discrimination modes. To augment the G matrix, the costs for taking measurements with the new discrimination modes was appended in at the end. The A matrix was created using the same procedure as before to fit the appropriate size.

A Bayesian network is a statistical model that represents a set of random variables and their conditional dependencies. To apply a Bayesian network model to the target classification, a directed acyclical graph (DAG) was created to define the probability that a target was threatening based on each discrimination mode (see Figure 6). In this DAG, each discrimination mode is assigned a probability chart. This probability chart outlines the odds of what each mode will be, based on the true target class. For instance, if the target is a threat, the target class will be 1. Then the probability of discrimination mode 1 being '0' is 20%.

3.5 Sensor Splitting

Many existing radar sensors are capable of 'beam-splitting', during which the radar can split its broadcasting power between two different viewing pixels. These separate beams provide measurements which are less accurate than a single focused beam, but the capability to measure multiple targets increases the flexibility of the system as a whole. For beam splitting applied



Target Class	Discrimination Mode 3: 0	1
0	0.8	0.2
1	0.3	0.7

Target Class	Discrimination Mode 1: 0	1
0	0.95	0.05
1	0.2	0.8

Target Class	Discrimination Mode 2: 0	1
0	0.8	0.2
1	0.2	0.8

Figure 6: Distributed Acyclical Graph Representing Conditional Dependencies of Target Class Based on Discrimination Estimates

within this simulation, it is assumed that total sensor power is divided evenly between each beam and a maximum of two beams are utilized. Each split beam can record either a state or discrimination measurement, but two beams from the same sensor may be configured to different measurement modes.

To account for beam splitting within the optimization, the number of combinations of sensors had to be expanded. Before, each sensor was required to be 'on', represented by a 1, or 'off', represented by a 0. This resulted in 2^{nS} sensor combinations, where nS is the number of sensors. Now sensors capable of beam splitting also have a third configuration that uses

half of the available power to record a measurement, represented by 0.5. Thus if nS_{split} sensors are capable of beam splitting, $2^{nS-nS_{split}} * 3^{nS_{split}}$ sensor combinations must be accounted for against each target. The cost function is then computed as before for all feasible sensor combination and target parings. To determine feasible sensor combinations, a reduced maximum range is used for sensors which are configured to utilize beam splitting in the combination.

The same constraints imposed on the original optimization must still be used for the new optimization problem. While the B matrix for optimization remains the same, the A matrix must be increased in size to fit the new sensor combinations. Still, the same algorithm for creating the A matrix can be used, as the only change is the number of combinations. Each assignment must still satisfy the constraint, imposed by $A * X = B$, that each sensor uses exactly the power available to it. This is a direct result of defining split sensors as 0.5 within the sensor combinations. If an assignment includes a split sensor, the split sensor must be used exactly twice on different targets. If an assignment includes no split sensors, each sensor must be used exactly once as before. No assignment can be made using a split sensor on one target and the same sensor unsplit on another target. The implementation of sensor splitting increases flexibility of the simulation, but also increases the required run time by requiring many more computations.

For sensors assigned to a beam splitting task, the assignment must now be calculated for each split beam. The existing architecture for storing assignments supported this capability. When the assignment involving beam splitting is passed to the measurement model, the model is configured to half

of the total sensor power and the range limit is reduced accordingly. The measurement is otherwise carried out as normal.

3.6 Dynamic Adjustment of Targets During Simulation

In the exising work, the simulations were limited to situations in which all targets are fired simultaneously. In reality, launches are unlikely to be perfectly synchronized, and other launches may be initiated after the initial volley. Sensors may also be unable to acquire certain targets until a time after they were launched, due to geometry or sensor limitations. Alternatively, it may be desirable to cease tracking of targets that have been intercepted or identified as non-threats. As such, the statically-defined number of targets in the previously existing simulation severely restricts the ability to test the applications of the software. To expand the applications of the software, code was added to allow targets to be added or neglected within the simulation after the initial volley is launched.

3.6.1 Adding Targets During Simulation

To implement the desired changes, modifications were required within the command station software and the allocation optimization software. The greatest challenges associated with this change were related to locating and updating all of the variables associated with the targets. From the approach taken in the existing work, some of these variables were initialized once and remained constant. These variables must be updated each time a target is added. Other variables, such as the covariance measurements, were initialized as a matrix or cell of null values with certain dimensions, and data

was inserted at each time step of the simulation. These variables had to be reallocated to make space for data related to the new targets.

Targets added to the simulation followed trajectories from a data structure which was created and saved prior to simulation. This data structure is the same one used to define all target trajectories for targets known at initialization, although any single target trajectory is limited to use by at most 1 target. It is important to recall that the weight function used to determine the optimal sensor allocation is dependent on the time until impact, and as such targets added much later than the initial volley may be viewed with less importance than potentially less-threatening targets that are much closer to reaching their impact location.

3.6.2 Removing Targets During Simulation

During a real-world scenario, it is highly unlikely that all targets will present a viable threat to any defended assets. Many military tools exist which are capable of intercepting hostile targets before they can cause harm to a defended asset. Even if a target is not intercepted, some targets may be destined for impact locations in open waters, clear of any defended asset. Also, other targets may be objects that are known to be non-threatening, including weather balloons. For any of these cases, these targets can be identified as nonthreatening through state and discrimination measurements. If a target can be conclusively identified as a non-threat, continuing to use sensor resources to refine the estimated state or discrimination will contribute very little benefit to overall situational awareness. However, continuing to account for this target would waste sensor time, as well as memory and

computation time within the allocation algorithm. As such, the ability to remove certain targets from the tracking provides a great benefit to the overall system.

To implement target removal, all variables associated with a target, besides its identification tag, must be removed. Still, it is desired to keep track of which targets were removed, which prohibits completely deleting these variables. To provide this functionality, the variables were simply replaced with NaN. The target identification is the only exception to this change, to ensure that knowledge of which target was removed is maintained in the simulation and in the data visualization tools. With this target removal function, plots will continue to update for all targets as usual, but future information for the removed target will be blank.

3.7 Software Modularization

A large push was made to segment the simulation into smaller blocks of code. The intent of this restructuring is to allow for a modular framework for the software. With this modular framework, certain segments of code could be substituted according to situation within the simulation. The form of the modular framework can be seen in Figure 7.

To demonstrate one benefit of a modular framework, consider the code which determines weighting of priority for sensor allocation. Under normal circumstances, sensors should be assigned targets which most improve the overall situational awareness; however, if a certain target is nearing its impact location that coincides with a defended asset, the target must be intercepted. There is a threshold uncertainty that must be maintained in

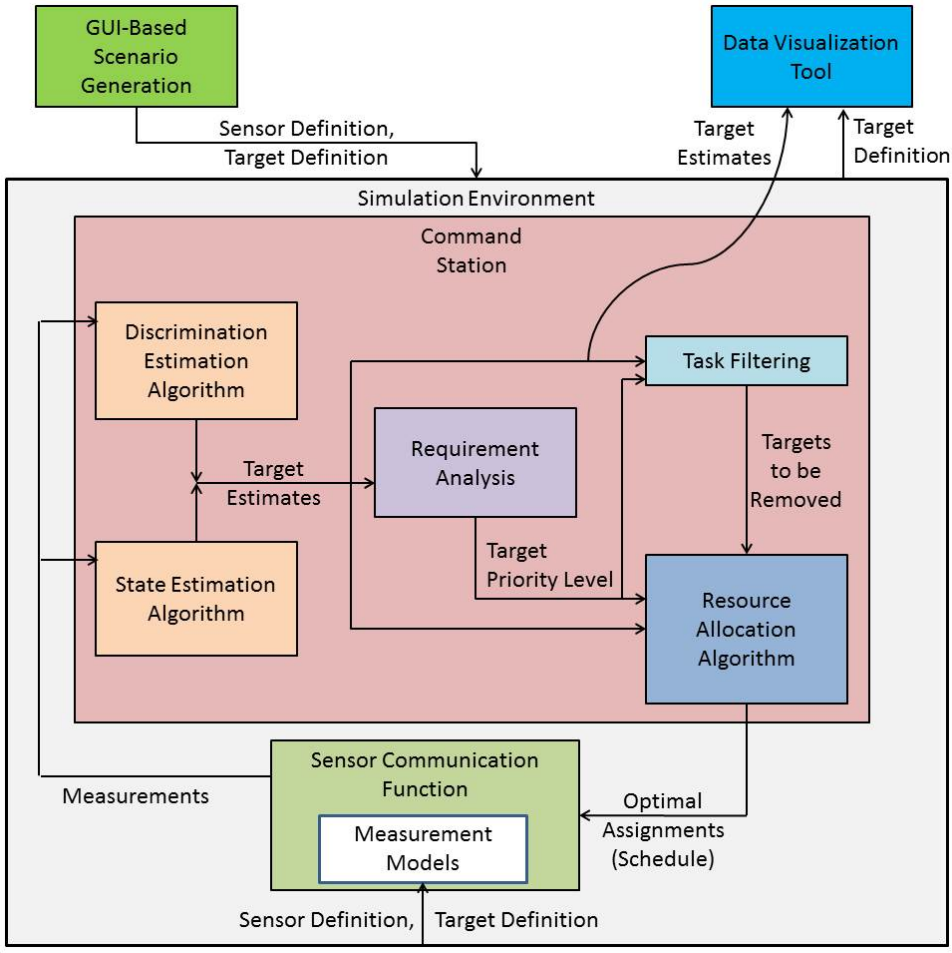


Figure 7: Representation of the Modular Framework for the New Simulation

order to successfully achieve interception. Here, it is assumed that without maintaining the threshold uncertainty limits the ability of the interception mechanism to accomplish its task. Therefore, sensor allocation should ensure the threshold is met even if overall situational awareness suffers some. Once interception has been completed, a sensor would also need to verify a successful interception. Modularized function calls allow for real-time switching of functions when necessary.

3.7.1 GUI-Based Scenario Generation

A new graphical user interface has been implemented to simplify the initialization of the simulation. This GUI loads saved target trajectory data, displaying the available trajectories within the Enemy Course of Action (ECOA) panel. The user can also create their own saved target data containing different ECOA trajectories. Then, the user selects how many targets should be drawn from each ECOA in the 'Raid Size Array' column, along with the time which those targets are launched in the 'Launch Time Array' column. If the user would prefer to simulate two separate volleys launched at different times using the same ECOA, the inputs into 'Raid Size Array' and 'Launch Time Array' can be entered as an array as shown. The number of target tracks available within each ECOA is limited, and the simulation will not allow multiple copies of the same target trajectory. To ensure the user does not exceed the limited number of trajectories for an ECOA, the number of unused tracks for each ECOA is displayed based on the current input settings. Once the user has selected the desired target trajectories from the ECOA panel, the trajectories are displayed on a map of the Earth.

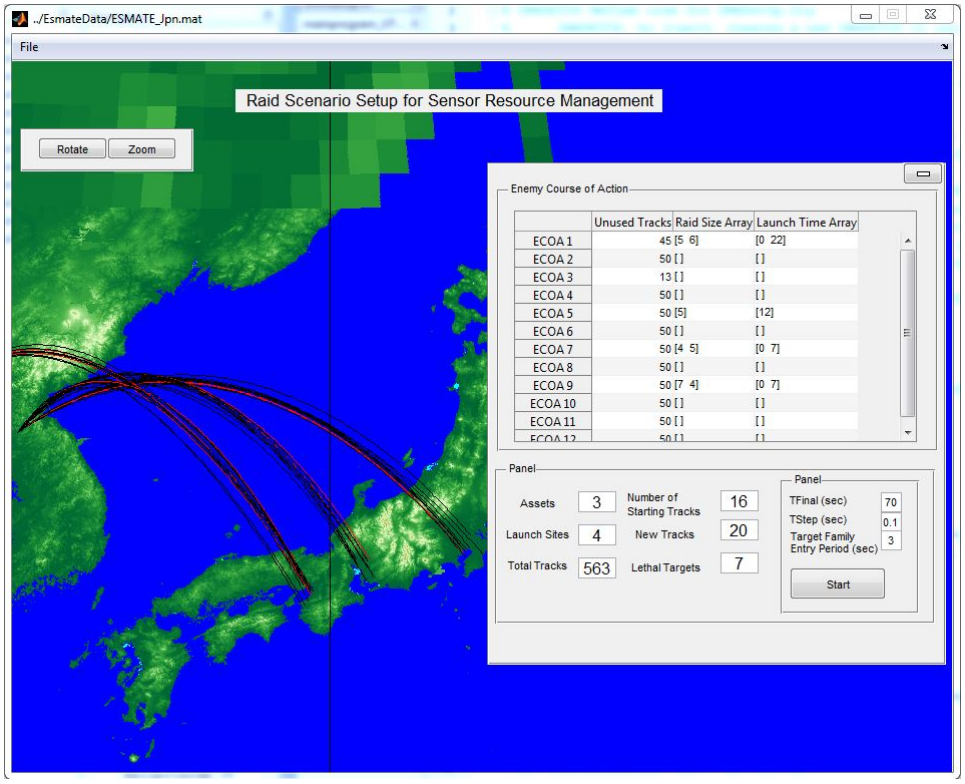


Figure 8: Graphical User Interface for Setup of Simulations

Targets shown in red are threats, whereas black trajectories correspond to non-threatening targets. The user can also configure the simulation time step length, the total simulation time, and the standard deviation of target launch around the specified launch times. Then the Setup GUI displays several parameters of the simulation, including the number of assets, the number of target launch sites, the number of initial targets, the number of targets added after initialization, and the number of lethal targets. Once the user is satisfied with the setup, they can save the configuration for later use and begin the simulation.

3.7.2 Data Visualization Tool

A new data visualization tool was also implemented to improve usability of the simulation (see Figure 9). This tool was made into a separate function to further modularize the simulation, creating plots using knowledge of the true target positions as well as target estimates. The tool creates a separate window containing all plots at the initialization of the simulation, and updates every 3 time steps. Seven plots are currently available, with the capability to add more if required. The plots can be minimized within the window to make room for other plots, or they can be maximized to take the full space in the window. Additionally, the number of plots shown at once can be configured using the 'Window Config' setting, including the options '1x2', '2x2', '2x3', and '2x4'. The plotting tool also gives the user the option to pause and resume the simulation while running.

The first plot (counter clockwise starting from top left) displays the true target impact locations compared to the predicted target impact locations at the current time step. Each prediction is connected to the associated true value using a blue line for clarity. Targets that are threats are represented in this plot with a larger circle for the true impact location and a larger, red asterisk for the predicted impact location.

The second plot displays the time history of the position covariance for individual targets. This position covariance is based on the EKF algorithm, and describes close each target estimate is expected to be from the true target. This plot is based entirely off of target estimate information, not true target information. The sixth plot displays the discrimination estimate

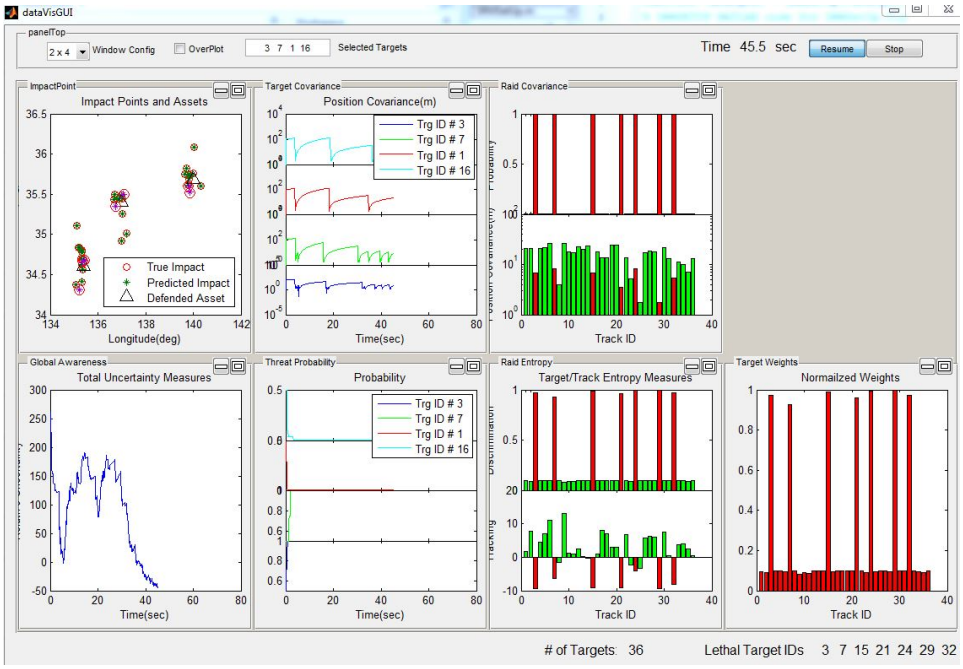


Figure 9: Data Visualization Window in 2x4 Configuration

for the same targets shown in the second plot. Again, this plot is based entirely off of estimated information, not true target knowledge. The targets shown can be modified by configuring the 'Selected Targets' option along the top ribbon. In addition to plotting each target individually within each plot, the option also exists to plot all targets on the same axes so that their magnitudes can be compared to each other. This feature is enabled by selecting the 'Overplot' option in the top ribbon.

The third plot shows the information used within the 'Covariance' optimization algorithm to calculate the cost functions. Two subplots are contained within this plot, with the first displaying the discrimination estimate for all simulated targets. Targets identified as threats are represented by a '1', and targets perceived as nonthreatening are signified by a '0'. The estimated covariance for all targets is depicted in the second subplot on a logarithmic scale. Targets that are known with more precision will have a lower covariance. Targets which are threats are identified by a red bar, while non-threats are green. The fifth plot similarly shows the information used within the 'Entropy' optimization algorithm to calculate costs. Again, two subplots are shown with the first depicting the discrimination estimate for all simulated targets. The second subplot displays the tracking entropy. It should be noted that these entropy values can become negative as a result of high-precision estimates, as the entropy calculation involves taking the logarithm of the determinant of the covariance matrix. Again, threatening targets are shown in red and non-threatening targets are green. Additionally, a list of the target numbers for all threatening targets is shown at the bottom right of the window under the label 'Lethal Target IDs'.

The fifth plot displays the weights assigned to each target. These weights are the same ones used for prioritizing assignments in the optimization algorithms. Since target weight is most heavily dependent on threat class of the target, it is expected that targets identified as threats will have weights near '1' while non-threatening targets will have weights closer to '0.1'.

The seventh plot displays the time history of the total uncertainty in Situational Awareness. This value does not have a physical meaning, but instead provides a general sense of how well the system is performing. A general downward trend should be observed as time progresses. It should be noted that the total uncertainty will spike upwards whenever a target is added to the simulation.

3.7.3 State and Discrimination Estimation Algorithms

To implement this modular framework, the code used to determine discrimination parameters was isolated from the main program and put into a separate function. The discrimination function receives the new measurements from the sensors for all discrimination modes available. It then computes the discrimination estimate for each discrimination mode and compiles these separate modes into a single discrimination parameter, just as before. Then the discrimination estimation algorithm returns only a single discrimination value ranging from zero to one for each sensor. By implementing the discrimination code in this way, it can be interchanged with new code in the future if a new discrimination model is developed. The state estimation algorithm uses the EKF to provide updates, and as such was already isolated into its

own algorithm. Thus no modification was needed to allow modularization of the state estimation algorithm.

3.7.4 Resource Allocation Algorithm

The newly developed resource allocation algorithm is a separate function from the other algorithms, just as it was for the existing simulation. This algorithm uses estimated target data to determine target weighting. Then, target data and weightings are used in conjunction with measurement models to determine the optimal pairing for sensors to targets. Finally, target estimated positions are used to define geometric assignments which can be interpreted and followed by each sensor. These geometric assignments ensure that the optimal pairing is executed.

The first task performed by the resource allocation algorithm is determining the priority level of each target. The target priority is used as a weighting for the assignment algorithm assignments, and is depicted in the target weight plot within the data visualization tool. Target weights are based entirely off of estimated target data, including state and discrimination estimates. The original weighting subfunction is based on target impact locations, with those expected to land near defended assets given higher priority. Any compatible weighting subfunction can replace the original impact-location-based cost function by changing a single line of code.

To demonstrate this capability, a new prioritization subfunction was implemented. The new subfunction utilizes a theoretical target phase, as well as the time remaining within the phase, to determine target priorities. Four phases exist in the current simulation. These phases are:

0. Initial Transient

1. Tracking and Discrimination

2. Interceptor Launch

3. Interception

The Initial Transient phase occurs directly after a target is acquired. During this phase, the command station should attempt to bring all targets below some maximum allowable covariance. Once this goal has been achieved, the Tracking and Discrimination phase begins. During this phase, assignments will prioritize discrimination of targets as long as all targets are within the maximum allowable covariance limit. If the target is classified as a non-threat, it will complete the rest of its trajectory in this phase. If instead the target is classified as a threat, the limits for the Interceptor Launch and Interception phases will be based on the requirements of the interceptor assigned to remove the threat (see Figure 10).

Each phase lasts for a given time period which can vary between all targets. This time period is based on the time at which the target is acquired and the target trajectory, with targets on more shallow trajectories spending less time in each phase. Each phase also is defined by covariance limits. For each phase, an upper and lower limit is assigned for the target covariance. It is desired that each target be below its lower covariance goal before the phase duration has passed. The lower covariance limit for any given phase matches the upper covariance limit for the following phase. For the Initial Transient and Tracking and Discrimination phases, these covariance

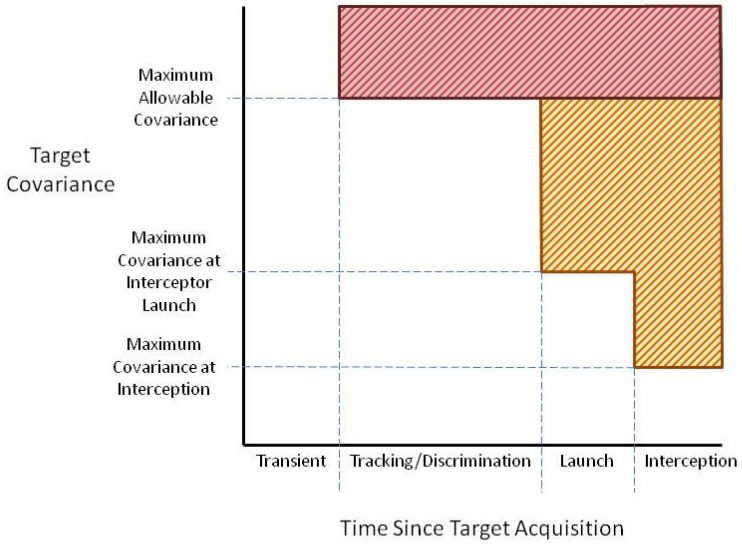


Figure 10: Phase Limits Based on Covariance and Time Constraints

limits are common for all targets. The covariance limits for the Interceptor Launch and Interception phases are ignored initially, as only targets which are discriminated as threats will enter into these phases. These limits are set whenever a target becomes classified as a threat.

The phase-based weighting function takes the form:

$$Wt_{state} = Wt_{min} + \frac{1 - e^{-\Delta x}}{1 + e^{\Delta T}}(Wt_{max} - Wt_{min}) \tag{3}$$

In this equation, Wt_{state} is the weight for a measurement of the state of a given target given that Wt_{min} is the minimum allowable weight and Wt_{max} is the maximum allowable weight. Δx is the difference between the actual phase x and the desired phase $xRef$. The desired phase is determined by comparing the duration of each phase, as determined through requirements analysis, to the time since target acquisition. The desired phase is an integer value. The actual phase is determined by comparing the target covariance to the limits for each phase. The actual phase is a real number determined by linearly interpolating between the covariance limits. Then, Δx is the highest value between 0 and $xRef - x$. Then, Δt is the time difference between the end of the actual phase and the current time. If the actual phase is behind the desired phase, it is possible for Δt to be less than zero. The result will be a sharp increase in the weight value for this target. Figure 11 depicts this weight function across various values of Δx and Δt for $W_{min} = 1$ and $W_{max} = 1000$.

When attempting to intercept targets, a limited window is available to launch the interceptors. Since discrimination of targets is only intended

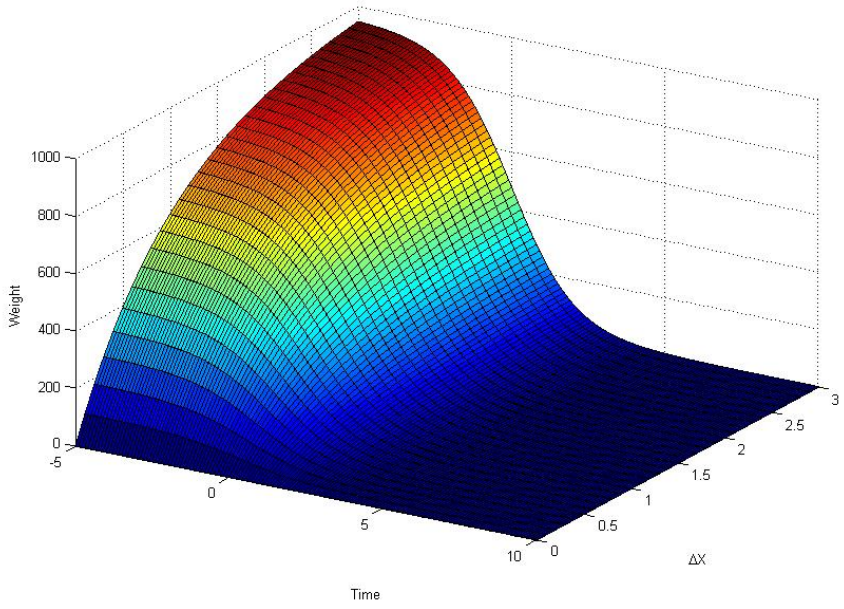


Figure 11: Phase-Based Weight Function Plot for State Measurements

to determine which targets must be intercepted, performing discrimination assignments after the Tracking and Discrimination phase has ended will not provide useful information. To ensure that the discrimination measurements occur while the desired phase is the Tracking and Discrimination phase, a different weight is assigned for the discrimination of targets. This discrimination weight is '0' for any target when $xRef$ is past this phase. Using this weighting ensures that no resources will waste time checking the threat level of a target which has passed its interception window. Before the Tracking and Discrimination phase begins, the weight for discrimination values is set to W_{min}. This ensures that discrimination will be considered during the Initial Transient phase, but only if all other target covariances are within the lower limit of this phase. During the discrimination phase, the weight for the discrimination of each target is calculated using Equation 3, except that $\Delta x = 0.9$. By using this equation, priority will be placed on the discrimination measurements unless $\Delta x > 0.9$, which occurs when a target is nearing the upper limit of its phase. The discrimination weight is then calculated using:

$$Wt_{dscr} = \begin{cases} 0 & xRef > 1 \\ W_{min} & xRef < 1 \\ W_{min} + \frac{1-e^{-0.9}}{1+e^{\Delta t}} & xRef = 1 \end{cases} \tag{4}$$

After the weight has been determined for each function, the optimal assignments are determined as before.

3.7.5 Requirement Analysis Algorithm

The requirement analysis algorithm provides the command station with the requirements for each target. These requirements may change based on the weighting function used, but may also be empty if no requirements are needed. For instance, the impact-location-based weighting function will determine weights based entirely off of the impact location, target class, and time until impact. No constraints are needed for this weighting function. Alternatively, the phase-based weighting function must be provided covariance and time constraints for each phase. This function provides these values. During the Initial Transient phase, no upper covariance limit is placed on the target. During the Interception phase, the lower limit is set to 0, although it is known that this value cannot be achieved. This limit is used to ensure that the high precision required for this phase is achieved.

3.7.6 Task Filtering Algorithm

The task filtering algorithm receives information about the target estimates and the target priority levels. With this information, the task filtering algorithm notifies the resource allocation algorithm of which targets may be ignored. Currently this algorithm is configured to remove targets with discrimination estimates below 0.02. When a target meets the criteria for removal, the task filtering algorithm calls the target removal function and keeps a record of the removed target.

3.7.7 Sensor Communication Function

The sensor communication function provides the measurements to the command station after receiving the optimal assignments from the resource allocation algorithm (see Section 3.3). The communication delay between the command station and each sensor is accounted for within this sensor communication function. This function uses true target and sensor knowledge from the simulation to determine measurements according to the measurement model function (see Section 3.2).

4 Future Work

While it has been shown that many improvements have been made to the simulation and SRM, there are still several areas in which the system could be improved. The simulation would greatly benefit from an improvement to its speed. MATLAB was used to implement the entirety of the simulation due to its ease of use. Still, it is known that the cost of using this easy-to-use tool is that run times can be much slower for code in MATLAB compared to other low-level languages, such as C++. In the future, some of the more time-intensive algorithms could be implemented in C++ or another similar language to improve the simulation run time. Even if another language is not used for implementation, there is still room for improvement in the speed of the simulation. Very little time was spent attempting to improve simulation run times, so it is possible that modifications to the simulation, even in MATLAB itself, could improve the speed of the simulations.

Improvements to the sensor scheduling could also be investigated. While the simulation is capable of producing a sensor schedule for several time steps, the schedule is still myopic and only considers the current time step when allocating the sensors in each step of the schedule. If non-myopic optimization is implemented, it is likely that an improved cost function must also be implemented, as the allocation algorithm is currently one of the more time-intensive components of the entire simulation.

In addition to these improvements, effort could also be spent increasing the available capabilities of the system. This ability to add new features is an intended consequence of the modularization of the code. New optimization

cost functions could be investigated and tested using the simulation model. Once they were tested, they could be evaluated against other cost functions currently available, at a variety of test cases, to determine which is more effective.

5 References

[1] Y. Bar-Shalom. *Multitarget-Multisensor Tracking.* Artech House, Boston, Massachussetts, 1990.

[2] A. Chhetri. *Sensor scheduling and efficient algorithm implementation for target tracking.* PhD thesis, Arizona State University, May 2006.

[3] A. Chhetri, D. Morrell, and A. Papandreou-Suppappola. Nonmyopic sensor scheduling and its efficient implementation for target tracking applications. *Applied Signal Processing*, 2006(1):1–19, 2006.

[4] P. Dodin, J. Verliac, and V. Nimier. Analysis of the multisensor multitarget tracking resource allocation problem. In *FUSION 2000*, volume 2 of *Information Fusion, 2000.* IEEE, 2000.

[5] C. Kreucher, K. Kastella, and A. Hero. Sensor management using an active sensing approach. *Signal Processing*, 85(1):607–624, 2005.

[6] V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden markov model sensors. *Signal Processing*, 50(6):1382–1397, 2002.

[7] A. Manne. A target-assignment problem. *Operations Research*, 6(3):346–351, 1958.

[8] MATLAB. *version 8.1.0 (R2013a).* The MathWorks Inc., Natick, Massachusetts, 2013.

[9] MATLAB and Optimization Toolbox Release 2013a. *version 8.1.0 (R2013a).* The MathWorks Inc., Natick, Massachusetts, 2013.

[10] J. Nash. Optimal allocation of tracking resources. In *Decision and Control*, volume 6 of *Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications*. IEEE, 1977.

[11] A Sinha, T. Kirubarajan, and M. Farooq. Performance evaluation of the randomized heuristic approach for multidimensional association. In *SPIE 5809*, volume 33 of *Signal Processing, Sensor Fusion, and Target Recognition*. SPIE, 2005.

[12] L. Stone, R. Streit, T. Corwin, and K. Bell. *Bayesian Multiple Target Tracking*. Artech House, Boston, Massachussetts, 2013.