# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
JUNE 1997

3. REPORT TYPE AND DATES COVERED
TECH REPORT

4. TITLE AND SUBTITLE
ASSESSING SITUATIONAL AWARENESS IN TASK FORCE XXI

5. FUNDING NUMBERS

6. AUTHOR(S)
E. TODD SHERRILL
DONALD R. BARR

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
OPERATIONS RESEARCH CENTER
USMA
WEST POINT, NY 10996

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

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

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Situational Awareness has become the central quest of U.S. Army force developments as the nation's ground combat arm of decision seeks to leverage greater effectiveness on the battlefield through information technology. As the term implies, situational awareness provides a combatant knowledge of his battlefield environment. A commander with complete situational awareness will know with certainty, among other elements of information, the status and disposition of his own forces as well as those of his opponent. Battlefield commanders throughout time have required some measure of situational awareness in order to impose their will on the enemy. Army leaders hypothesize that information age technology can be used to achieve information dominance over the enemy and that units equipped with greater situational awareness will fight more successfully than units without the added capability.

In an effort to test this hypothesis the Army conducted an Advanced Warfighter Experiment (AWE) which began at Ft. Hood, TX and culminated in a focused rotation at the National Training Center, Ft. Irwin, CA. Although many initiatives in the area of information dominance were tested in the AWE, the centerpiece of the program was a test case unit designated as Task Force Twenty-One (TF Xyd). TF XXI was a normal heavy maneuver brigade out of Ft. Hood, TX. The Army equipped and trained TF = with the most promising prototype technology designed to provide commanders real-time situational awareness and information dominance. TF XXI was then tested against an opposing force at the NTC in live simulated combat.



14. SUBJECT TERMS
TASK FORCE XXI

15. NUMBER OF PAGES
72

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 | |

# ASSESSING SITUATIONAL AWARENESS IN TASK FORCE XXI

**MAJ E. Todd Sherrill**
**Operations Research Center**

**and**

**Professor Donald R. Barr**
**Department of Systems Engineering**

**U.S. Military Academy**
**West Point, NY 10996**

A Technical Report
of the
Operations Research Center
United States Military Academy

Directed and Aproved by
LTC Michael L. McGinnis, Ph.D.
Director
Operations Research Center

June 1997

## INTRODUCTION

*Situational Awareness* has become the central quest of U.S. Army force developments as the nation's ground combat arm of decision seeks to leverage greater effectiveness on the battlefield through information technology. As the term implies, situational awareness provides a combatant knowledge of his battlefield environment. A commander with complete situational awareness will know with certainty, among other elements of information, the status and disposition of his own forces as well as those of his opponent. Battlefield commanders throughout time have required some measure of situational awareness in order to impose their will on the enemy. Army leaders hypothesize that information age technology can be used to achieve information dominance over the enemy and that units equipped with greater situational awareness will fight more successfully than units without the added capability.

In an effort to test this hypothesis the Army conducted an Advanced Warfighter Experiment (AWE) which began at Ft. Hood, TX and culminated in a focused rotation at the National Training Center, Ft. Irwin, CA. Although many initiatives in the area of information dominance were tested in the AWE, the centerpiece of the program was a test case unit designated as Task Force Twenty-One (TF XXI). TF XXI was a normal heavy maneuver brigade out of Ft. Hood, TX. The Army equipped and trained TF XXI with the most promising prototype technology designed to provide commanders real-time situational awareness and information dominance. TF XXI was then tested against an opposing force at the NTC in live simulated combat.

Since the authors had demonstrated the utility of an information measure which we call information gain, OPTEC asked us to attempt to apply the measure to data generated during the AWE trials [1][2][3] [4] [6] [7]. In this paper we report our application of the information gain measure as well as a fratricide measure which we have termed the fratricide index. We applied these measures to data collected during TF XXI's train-up at Ft. Hood. We specifically report on data generated during company level training.

In order to calculate these measures we developed a data reduction tool using Microsoft Access version 7.0 software. Access provided us the query power of an object oriented relational data base along with the capability of structured programming. Appendix C contains the code behind our implementation with appropriate documentation of the objects and algorithms involved. The code is written in Visual Basic.

## THE INFORMATION GAIN MEASURE

Information gain measures the Blue forces' awareness over time of Red's disposition. For our purposes disposition means number and location of Red combat systems such as tanks and armored personnel carriers. Within a time interval of duration $\Delta t$, say $(t, t+\Delta t)$, the measure is a distance measure between two probability distributions

$P_t$ and $P_{t+\Delta t}$ which we refer to as the prior and posterior distributions respectively. These distributions represent the discrete probabilities, from Blue's perspective, that a Red vehicle is in various areas of the battlefield. Consider the case of one enemy vehicle located somewhere on the battlefield. If the battlefield were partitioned into cells each cell would have a certain probability of containing the Red system. The sum of the discrete probability values over all cells would be 1.0 with those areas of greatest likelihood having the larger values. At the beginning of the time interval (t, t+$\Delta$t) Blue's uncertainty about the Red disposition is represented by the prior distribution $P_t$. If the Blue force believes that the Red vehicle is equally likely to be in any one of the cells, the prior distribution would be uniform over the cells.

When any Blue sensor scans an area of the battlefield Blue gains information about the enemy disposition. Assuming a perfect sensor, Blue will either determine Red's location or Blue will discover cells where Red is not located. The magnitude of the new information depends on the operating characteristics of the Blue sensor as well as the outcome of its scan. For example, if a particular Blue sensor has a probability of detection ($P_D$) of .8 then this same sensor has a .2 probability of failing to detect a target's presence in a scanned cell. The cells searched by Blue during the interval (t, t+$\Delta$t) receive updated probability assignments based on the operating characteristics of this sensor. Our method of updating the probability distribution from $P_t$ to $P_{t+\Delta t}$ is an application of Bayes' formula [1]. The Bayesian calculations incorporate $P_D$ and $P_t$ values in order to update to the posterior distribution $P_{t+\Delta t}$. This posterior distribution represents Blue's new uncertainty about Red's disposition and becomes the prior distribution for the next time step, (t+$\Delta$t, t+2$\Delta$t).

The prior is updated to the posterior using knowledge of which cells have been searched and the $P_D$ of the searching sensor(s). Let T(j) denote the event that there is an enemy vehicle in cell j and let I(j) denote the event that Blue sensors report that there is an enemy vehicle in cell j. If we assume zero false alarm rate for Blue sensors we have:

$$P[T(j) \mid I(j)] = 1.0;$$
$$P[T(i) \mid I(j)] = 0.0; \quad \text{where } i \neq j;$$
$$P[T(i) | \sim I(j)] = \frac{p_i}{1 - P_D p_j}; \tag{1}$$

and

$$P[T(j) | \sim I(j)] = \frac{(1 - P_D) p_j}{1 - P_D p_j}, \tag{2}$$

where, "*~I(j)*" indicates the event "search in cell *j* fails to detect the target. Equations (1) and (2) apply to the situation where Blue searches and fails to detect the target during one time interval. Equation (1) applies to a cell where Blue does not look. Equation (2) applies to a cell where a Blue sensor looks and fails to detect. The original probability assigned that cell ($p_j$) is reduced but is not driven to zero unless the $P_D$ of the Blue sensor is 1.0 The $p_i$ and $p_j$ in equations (1) and (2) above refer to these individual cell probabilities. Since the denominators in each case are identical we treat them as a multiplying constant. If Blue finds the target, the cell containing the target is assigned a cell probability of 1.0. All other cells are assigned zero probability since Blue knows the vehicle's location. See [1] for a complete development of this formulation.

As mentioned above, information gain is a measure of the distance between the prior and posterior distributions. This distance is represented as the change in *entropy* resulting from updating the prior to the posterior distribution. Shannon defined entropy as a measure of randomness or uncertainty [5]. For our application the entropy (uncertainty) of the posterior distribution is subtracted from the entropy of the prior distribution. In this respect the metric captures the decrease or increase in uncertainty concerning the location of Red systems during each time interval. This change in entropy is information gain:

$$\delta(p_t, p_{(t+\Delta t)}) = \sum p_{(t+\Delta t)} \ln(p_{(t+\Delta t)}) - \sum p_t \ln(p_t),$$

where summation is over all cells for which $P_t$ ($P_{t+\Delta t}$) is positive [1].

## Example

Assume the Blue sensors are perfectly accurate (i.e., in each cell searched, $P_D$ = 1.0 and false alarm probability is zero). If Blue detects the Red vehicle in cell j, then 1.0 is assigned to cell j and zero probability is assigned to all other cells. Blue's cumulative information gain will be at maximum value since Blue now knows all there is to know about this Red vehicle. In the case of one vehicle located in one of 100 cells, the maximum amount of information that could be attained is ln(100) = 4.605 [1]. If the enemy vehicle is detected during the 1st time step, the information gain for that step would be the maximum value and the search would be over. Likewise, the search is over when the vehicle is detected during any time step and the information gained, for this time step, is the maximum possible gain (4.605) minus the cumulative gain up to the time of detection.

When Blue searches for multiple Red vehicles we simply multiply, at each time step, the information gain for one vehicle by the number of Red vehicles. In our example, assuming five enemy, the maximum gain would be 5*ln(100) = 23.026. When we search a cell and find no vehicles we know that none of the five vehicles is in that cell, hence five times the gain for an individual vehicle. If we find a vehicle during the search, the information gain concerning that particular vehicle makes a jump in value up to ln(100) or up to one fifth the total possible gain. For those vehicles remaining undetected, the gain generated by searching and not finding is now multiplied by four; we have found where four vehicles are not located. Figure 1 illustrates this approach. The graph at the right of Figure 1 represents the sum of the two plots shown in the leftmost graph.

We transform the information gain values to the scale [-1,1] so that the values calculated over each $\Delta$t are relative to how much information could be known. We divide $\delta(p_t, p_{(t+\Delta t)})$ by the maximum entropy. This gives us a normalized scale and a basis for comparison.

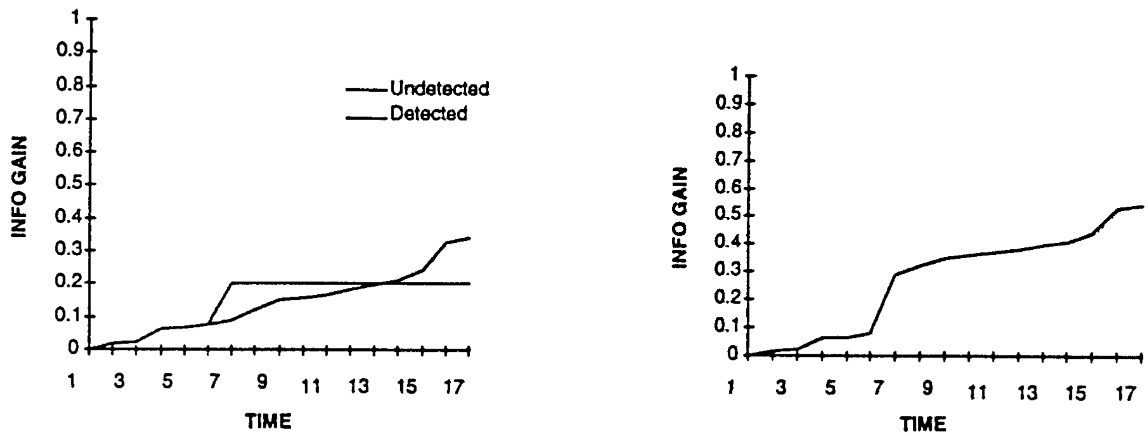



Figure 1. Information gain over time for one detected & four undetected vehicles is shown in the graph on the left. Detection occurred during time period 6. The graph on the right shows the cumulative information gain over time about all five enemy vehicles. Values are scaled to [0,1] as discussed above.

Though the theory is rather simple, its implementation can be very challenging. The Bayesian formulation above requires three types of data during each time step: 1) knowledge of which areas of the battlefield (cells) Blue sensors looked in, 2) the probability of detection ($P_D$) for the sensors that did the respective scanning, and 3) the prior $P_t$. Since information gain credits finding where the enemy is not, we need to know at each time increment what terrain cells Blue sensors have searched regardless of the presence or absence of enemy vehicles. Our approach to addressing these challenges is discussed below.

## APPLICATION

One of the central objectives of the AWE was to ensure participating units were adequately trained in the use of new equipment prior to the NTC trials. Accordingly, Task Force XXI conducted individual and collective training with the new systems at Ft. Hood, Texas. OPTEC collected data from these training trials at Ft. Hood in order to "shake-out" the data-collection plan and to begin to gain insight into the value of digitization. Company Team Situational Training Exercises (STX)[1] provided the first opportunity to begin to assess the effects of digitization. The plan called for some units to conduct STX without any of the prototype situational awareness technology (SAT). Other units would conduct STX with SAT allowing analysts the opportunity to assess differences. The company team STX scenario required the training unit (Blue) to conduct an offensive operation against a defending (Red) force.

[1] A Situational Training Exercise is an attempt to replicate combat conditions for the training unit. The force-on-force exercise is designed to stimulate planning, execution and assessment of unit combat operations and can be orchestrated to force the execution of particular battle drills.

Our initial challenge was to determine if the data collected during Company Team STX could be used to compute Information Gain. OPTEC provided a sample data set for this purpose. The data set contained order of battle, Red kills (time of death for Red Vehicles), Blue detections (time and location of Red vehicles as reported by Blue), and location information concerning the positioning of all vehicles (Red & Blue) over time. Table 1 shows an example of location information.

| MISSNUM | DATE | PID | PLAYER | STIME | ETIME | POSITION |
|---|---|---|---|---|---|---|
| 019 | 960416 | 261 | B21T 3/C/2-8 | 081957 | 082519 | PK17[illegible]2588 |
| 019 | 960416 | 261 | B21T 3/C/2-8 | 082519 | 082717 | PK1777598 |
| 019 | 960416 | 261 | B21T 3/C/2-8 | 082717 | 083012 | PK1807604 |
| 019 | 960416 | 261 | B21T 3/C/2-8 | 083012 | 083148 | PK1779619 |
| 019 | 960416 | 261 | B21T 3/C/2-8 | 083148 | 083204 | PK1739621 |
| 019 | 960416 | 261 | B21T 3/C/2-8 | 083204 | 083230 | PK1731623 |

Table 1. Example location data for Company Team STX.

The Blue detections were composed of both Blue calls-for-artillery-fire on suspected enemy vehicles and Blue spot reports concerning Red's disposition which Blue conveyed over the unit command net. Table 2 shows a sample of Blue detections.

| ID | MSN | TIME | LOCATION | ENEMY | SOURCE |
|---|---|---|---|---|---|
| 52 | 260 | 21OCT96:14:29 | 06605150 | T80 | Spot |
| 53 | 260 | 21OCT96:15:41 | 07305230 | T80 | Spot |
| 57 | 240 | 17OCT96:12:47 | 07205230 | BMP | Fs |
| 58 | 240 | 17OCT96:12:47 | 07205230 | T80 | Fs |

Table 2. Example detection data for Company Team STX

As mentioned earlier, the information gain MOE is computed for each prior and posterior probability distribution of Red positions over some time interval $\Delta t$. The key to successfully applying the measure is to adequately represent $P_t$ and the inherent updating to $P_{t+\Delta t}$ that must take place over a given time period. Accounting for information gain due to detections or kills of enemy vehicles is relatively straight forward. Accounting for information gain due to unsuccessful searching is more challenging. In order to implement the measure we made two assumptions which we discuss below.

Equations (1) and (2) above were originally developed as a result of our efforts to automate the information gain measure in the Janus wargame [7]. In combat simulations the analyst has the ability to establish and view the entire battle area. In such situations it is possible to speak in terms of individual *cells* of the battlefield holding some relative probability of containing an enemy vehicle.

We cannot manage a cell-by-cell updating of probabilities in live training exercises such as Company STX. While units are restricted to a "maneuver sandbox," representing terrain cells and determining which cells were searched through time would be logistically and computationally prohibitive. Also, the sample STX data provided by

OPTEC did not seem to support such an undertaking. Our measure, however, demands that we know where Blue has looked during each time period.

We settled upon a boolean approach much like that employed in the board game Battleship. In the board game the attacker begins with a uniform distribution of his opponent's positions. As he fires salvos he receives information about his opponent's positions in the form of a boolean variable (hit or miss). For the sake of this discussion let's assume for a given salvo, Blue receives a "miss." The attacker has eliminated one cell but his knowledge of his opponent is still uniformly distributed over the remaining cells. There is greater probability that a ship is in one of the remaining cells since it was not found in the cell just searched.

Relating the board game discussion to our Bayesian formulation, we assume that the $P_D$ for the attacker is 1.0 and the original prior distribution is assumed to be uniform over the set of 100 cells. Thus, recalling equation (2), the $(1-P_D)$ factor updates the likelihood that an enemy vehicle is located in a searched cell to zero probability. In this case, the probability that was originally represented in the searched cell is distributed proportionally among the remaining cells. Since the remaining cells began with equal values the Bayesian updating process will equally distribute the added probability among these cells. The posterior distribution will, therefore, also be uniform over the smaller set of cells remaining. In Figure 2 we show a visual representation of a uniform discrete bivariate distribution over a battle area (the size of the Battleship grid) that has been partially searched by Blue forces. Blue forces searched along the route illustrated in Figure 3.

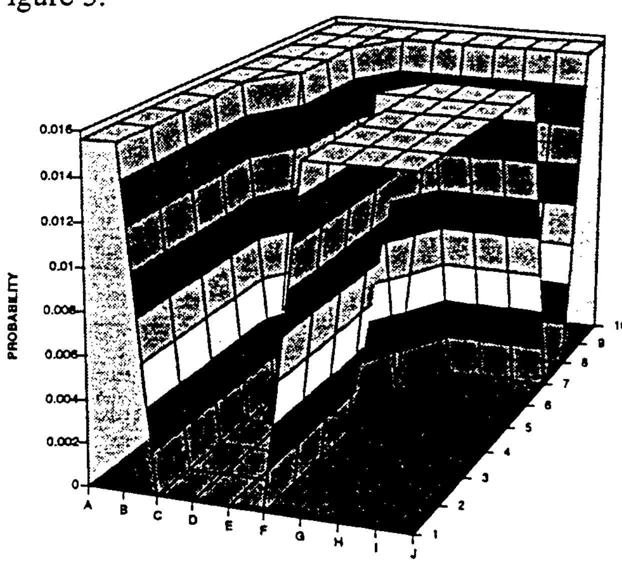



Figure 2. Posterior distribution of Red's likely location following Blue's search.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | 10 |
| | | | | | | | | | | 9 |
| | | | | X | X | X | X | X | X | 8 |
| | | | X | X | X | X | X | X | X | 7 |
| | | | X | X | | | | | | 6 |
| | | X | X | X | X | | | | | 5 |
| | | X | X | X | X | | | | | 4 |
| | | X | X | X | X | | | | | 3 |
| | | X | X | X | X | | | | | 2 |
| | | X | X | X | X | | | | | 1 |
| A | B | C | D | E | F | G | H | I | J | |

Figure 3. Path searched by Blue Forces in Figure 2.

The $p_i$ values were originally .01 since the grid has 100 cells. Notice in Figure 2 how the probability values of the searched cells have decreased to zero and the probability values for the remaining cells have increased to .016. The smoothing in the graph is simply a function of the graphical software and is not reflected in the calculated probability levels.

The uniform distribution assumption enables us to keep track of searched areas without trying to maintain a cell-by-cell accounting. If we were to partition the battlefield area (BA) into cells of fixed size ($A_c$) then, given a uniform prior, the $p_i$ for each cell would be 1/(#cells in area BA). Consider a unit searching an area Z during some $\Delta t$ such that $Z \subseteq BA$. Assuming the unit's sensors have a $P_D$ of 1.0, we are left with posterior values $p_i^* =$ 1/(#cells in area (BA-Z)). Recall that information gain is simply the change in entropy during each interval of time $\Delta t$. So we have:

$$\delta(p_t, p_{(t+\Delta t)}) = \sum p_{(t+\Delta t)} \ln(p_{(t+\Delta t)}) - \sum p_t \ln(p_t)$$

$$= \ln(\#cells(BA - Z)) - \ln(\#cells(BA))$$

$$= \ln\left[\frac{\#cells(BA - Z))}{\#cells(BA))}\right]$$

Since $\#cells(BA) = \dfrac{area(BA)}{area(A_c)} = k * area(BA)$ we can simplify the computation of information gain to $\ln\left[\dfrac{area(BA - Z)}{area(BA)}\right]$. This adaptation of the theory reveals that we simply need to determine the amount of battlefield area Blue has searched in order to compute the information gained through searching.

## ESTIMATING AREA SEARCHED

The location data provides us *ground-truth* about Blue's disposition over time. We use this information to compute movement of the Blue unit's center of mass (CM) throughout the battle. We calculate CM at times t and t+$\Delta$t using the eight digit grid coordinates of each Blue vehicle. We then compute vectors extending from $CM_t$ to $CM_{t+\Delta t}$ which, taken in sequence over the duration of the operation, represent the unit's movement throughout the battle. A unit moving forward in the offense will sweep-out (search) a path through the area of operations. We assume that the Blue unit has searched this swept area completely (so $P_D = 1.0$). In order to determine the amount of area searched we estimate the spread of the unit. This estimate is the distance (*d*) between the vehicle farthest from the CM in the positive normal direction and the vehicle farthest from the CM in the negative normal direction. We add 1500 meters to this distance to represent the ability of these peripheral vehicles to scan beyond their physical locations. For each time step, the area searched is the length of the movement vector (*l*) times the distance of spread (*d*). Note the set of *l* values gives insight into the Blue unit's movement rate. Figure 4 illustrates this model.

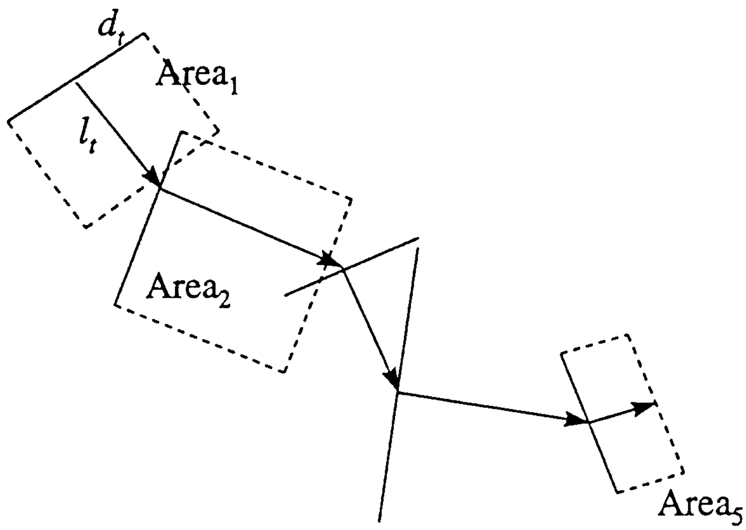



Figure 4. Model of Area searched during each $\Delta t$. Area$_t$ = $d_t l_t$ where $l_t$ is the magnitude of the movement vector and $d_t$ is the spread of the unit normal to the direction of travel.

Figure 4 makes it evident that some area is double counted and some area is not counted. Since the Blue units generally moved in a direct route from line of departure (LD) to the objective area, we felt that ignoring the overlaps and shortfalls would be acceptable for our purposes.

The curve in Figure 1 labeled "undetected" is an example of information gain due to searching. This curve shows how the information gain measure credits the searching unit for learning where the enemy is not located.

## INFORMATION GAINED THROUGH DETECTIONS

Recall the sample detection data of Table 2. Each reported detection consists of a detection time and an eight digit grid location of the detected Red vehicle. We use this data and exploit the assumption of uniform distributions to calculate the information gain due to Blue detections of Red vehicles. Naturally, Blue should gain considerably more information from finding a vehicle than he does from finding where the vehicle is not. The spike in gain during time period 6 of Figure 1 is representative of this difference.

Our approach to detection is essentially the same as our approach to measuring information gain due to search. When Blue detects a Red vehicle the distribution of Red's location from Blue's perspective is updated so that all of the probability is over the spot on the battlefield where Red was found. Zero probability is over the rest of the battlefield, for that vehicle. With probability 1.0 Red is somewhere within the area where Blue detected him. The size of this area and hence the accuracy of Blue's detection determines the magnitude of the information gained. The smaller the area the greater the gain.

We use circular area to measure the amount of information gained due to a detection. Blue will seldom be exactly correct in his estimate of Red's location. Let $M_d$ be the distance between Red's actual location (eight digit grid) and the eight digit grid where Blue reports Red to be. We credit Blue with locating Red within a radial distance of $M_d$ which we term the radial missed distance or $RM_d$. Blue has narrowed the search for Red to an area of the battlefield of size $\pi(RMd)^2$.

In effect we have a qualitative measure of the Blue detection. Some detections provide Blue with more information than others. In the sample output that we show below, spikes of information gain reflect detections. The varying sizes of these spikes reflect the respective accuracy of the Blue detections.

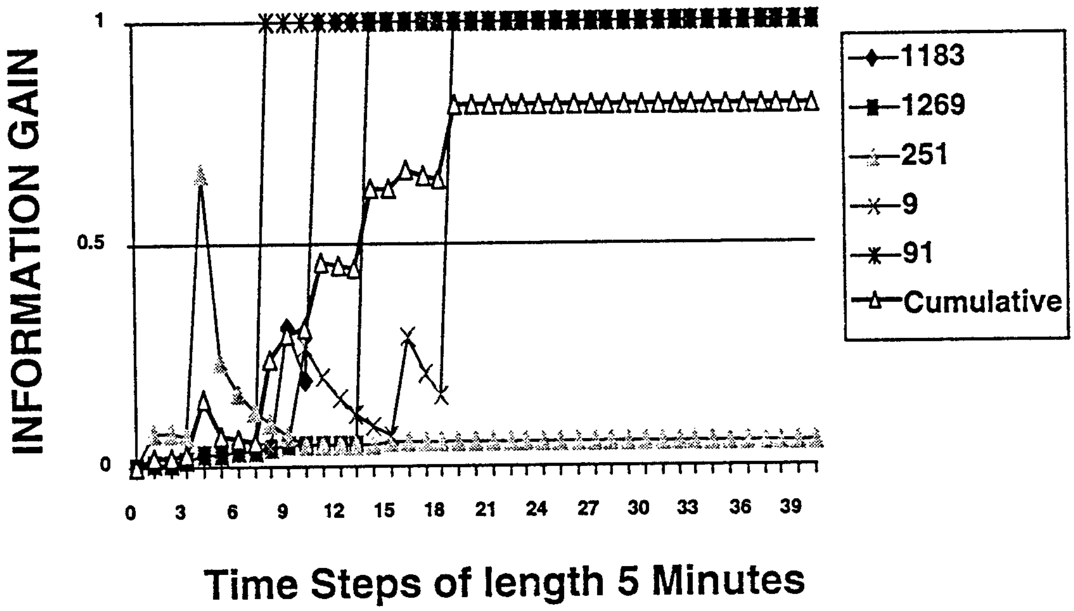



Figure 5. Information gained about each enemy vehicle over time by a Blue Company Team in the attack, normalized to the interval [-1,1].

Note also in Figure 5 that the spikes due to detections degrade over time. We felt that a degradation effect was necessary to realistically model Blue's situational awareness since information is so extremely time sensitive. If a detected Red vehicle is not killed or re-detected we allow the size of the circular area computed at the time of detection to expand uniformly over time. Blue's spike of certainty "melts" with each passing time period as the size of the area possibly containing the Red vehicle grows. The rate of degradation is determined by the likely movement rate of Red vehicles. We assume an average movement rate of 3 kilometers per hour for Red systems since, in the STX scenario, Red is defending.

Figure 5 also shows the cumulative information gain for Blue over all enemy systems. Computing the total information gain occurring during a time step requires a summation of the varied contributions to the total from each individual enemy system. We therefore must keep account of the state of each enemy system from Blue's perspective. The possible states are: 1) Area - Blue is searching and finding where the vehicle is not located, 2) Detection - The vehicle has been found, 3) Degradation - The vehicle was detected but not killed, 4) Kill - The Blue force has killed the Red vehicle. Enemy vehicles transit from one state to another at the conclusion of a time step. The possible transitions are depicted in Figure 6.

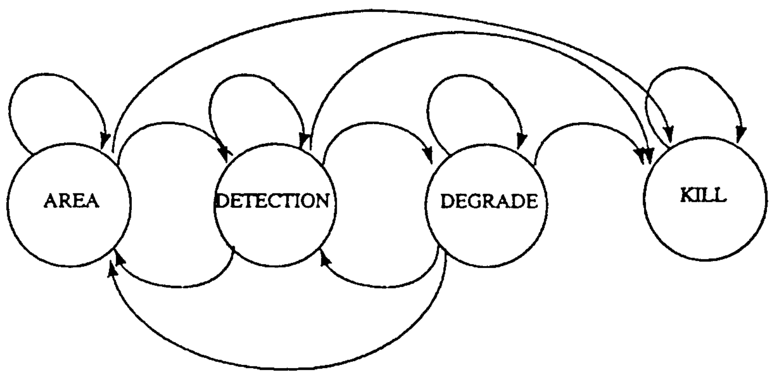



Figure 6. State transitions of enemy vehicles from Blue's perspective.

All enemy vehicles begin in the Area state. When enemy vehicles are in the Area state, Blue information gains are determined by Blue searching and eliminating possible locations of these enemy vehicles.

Vehicles in the Detection state have been detected by at least one Blue sensor. Blue gains substantial information from a detection as can be seen in Figure 5.

Degradation begins in the time step immediately following the time step in which detection occurred provided the vehicle is not detected again or killed.

When Blue kills an enemy, Blue knows all there is to know about that enemy. We assume that this information does not decay. The dead vehicle's contribution to Blue's situational awareness reaches and remains at maximum value. This is illustrated in Figure 5 for vehicle 91 at time 7, vehicle 1183 at time 10, vehicle 1269 at time 13, and vehicle 9 at time 18.

Note that it is possible to transit directly from the Area state to the Kill state. This may seem counterintuitive. It happens when a particular vehicle is detected and killed during the same $\Delta t$. Note also that a vehicle can transit from Degrade to Area. This occurs when the spike in information gain due to detection has degraded over time to the point that no more is known about this particular vehicle than is known about those vehicles that are in the Area state. Likewise a vehicle can transit from Detection directly to Area. This happens when a vehicle in the Detection state is detected again but with such poor quality that the size of the circular area possibly containing the Red vehicle is greater than the remaining area of the battlefield to be searched.

In this regard the information gained through searching and not finding serves as a lower bound on degradation. Vehicle 251, in Figure 5 above, degraded down to this lower bound after being detected by Blue with great accuracy. Vehicle 251 remained in the Area state for the rest of the battle. Blue's only awareness of vehicle 251, after this initial detection, was gained by finding where 251 was not located.

As Table 2 reveals, we do not know which enemy vehicle is detected when a detection occurs. The detection data contains only the type enemy, the time of detection (TOD), and the location reported by Blue. In order to determine the identity of the enemy vehicle we employ the query capabilities of our data reduction tool. We query the location table for the nearest enemy vehicle to the reported grid location within a time window of TOD to TOD minus 5 minutes. The five minute window allows for transmission time of unit spot reports and calls for artillery fire. We treat this nearest vehicle as the detected vehicle and compute the circular area as described above.

Knowing that Blue sensors could mistakenly report Blue vehicles as enemy, we felt it would be of interest to seek insight into the potential of such mistakes. We developed an index to reflect the potential of fratricide for each reported detection. We hoped that this measure would allow us to compare fratricide in baseline and treatment trials.

## FRATRICIDE INDEX

The fratricide index provides a qualitative measure of friendly spot reports concerning enemy vehicles. As its name implies, the measure indexes the accuracy of spot reports relative to the possibility that the reported enemy vehicle is actually a friendly vehicle. Inputs for the measure are the reported location of an enemy vehicle (spot report), the actual location of the enemy vehicle closest to the reported location, and the actual location of the friendly vehicle closest to the reported location. The measure ranges from -1.0 to 1.0 with 1.0 signaling minimal possibility of fratricide and -1.0 signaling maximum possibility of fratricide.

$$FI = \frac{|f|-|r|}{d}$$

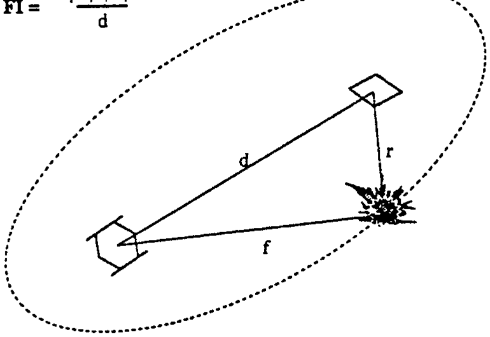



Figure 7. Fratricide Index

A graphical and mathematical model are presented in Figure 7. In the graphical model the reported location of the enemy vehicle is represented as an artillery strike. The length $r$ is the distance from the reported location to the nearest enemy vehicle at the time of the detection. Likewise, the length $f$ is the distance from the reported location to the nearest friendly vehicle at the time of detection. The time of detection is treated as a time window of five minutes which begins five minutes prior to the detection time (the time that the spot report was received). In other words, which vehicles were closest to this reported location within five minutes prior to the spot report? The five minute window can be thought of as the command and control ($C^2$) lag time of the report. The length $d$ is the distance between the friendly and enemy vehicles.

When $f$ is greater than $r$ the quality of the report is indexed between 0 and 1.0 as shown in Figure 8. When $f - r = d$ which is to say the reported location is either exactly correct or safely on the far side of the red vehicle from blue's perspective, the possibility of fratricide is low and the quality index is 1.0. See Figure 9. Figures 11 and 12 show less favorable fratricide situations which correspond to poor report quality and negative index values. Figure 10 represents situations in which the reported location is closer to Blue than Red ($f$ is less than $r$ ). Figure 11 represents the greatest possibility of fratricide and hence the lowest possible quality measure of -1.0. Finally, Figure 12 represents the "middle of the road case" of FI = 0.0 which occurs when $f = r$.

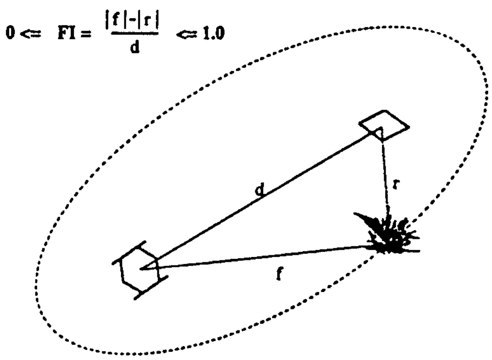



Figure 8. Reported location is closer to Red than Blue.

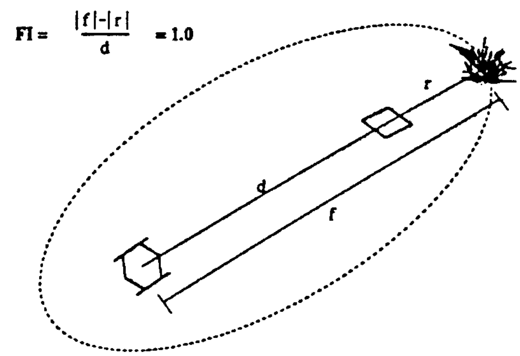



Figure 9. Reported location is either exact or safely beyond the target.

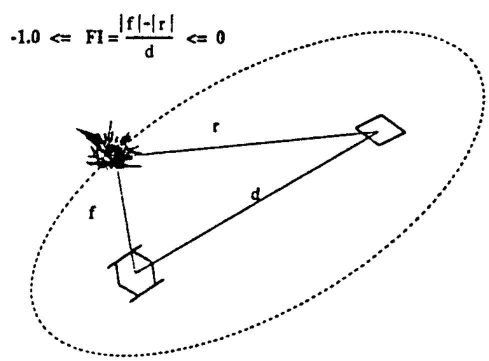



Figure 10. Reported location is closer to Blue than Red.

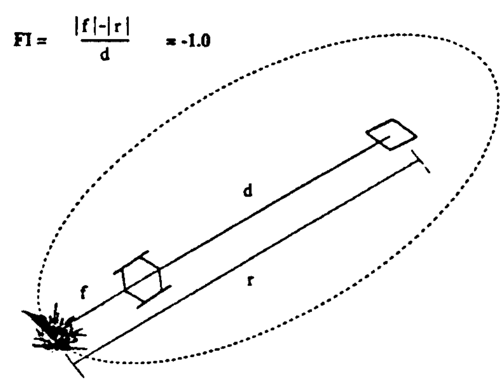



Figure 11. Reported location is either directly on or dangerously behind Blue.

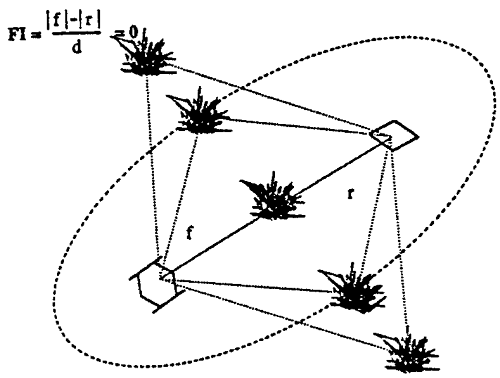



Figure 12. Reported location is equal distance from Blue and Red.

## CONTOURS OF THE FRATRICIDE INDEX ARE HYPERBOLAS

We believe some insight into the fratricide index is gained by noting that the index takes on equal values along hyperbolas in the plane. By definition, a set of points H is a *hyperbola* if and only if there are two points $F_1$ and $F_2$ and a positive constant a such that, for every point $A \in H$,

$$| d(A, F_1) - d(A, F_2) | = 2a, \tag{3}$$

where d is the Euclidean distance function [8]. The points $F_1$ and $F_2$ are called the *foci* of the hyperbola. For our application, the foci are the locations of the Blue and Red vehicles involved in determining the distances f, r, and d, as we shall see.

Suppose the foci are at the points (-c,0) and c,0) on the abscissa of a Cartesian coordinate system. Let $b^2 = c^2 - a^2$. The equation of the hyperbola is then given by

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1. \tag{4}$$

Now consider the defining relationship, $(f - r)/d = i$, for a fixed value i, $i \in [-1, 1]$, of the fratricide index. This relationship can be expressed in the form of Equation (3), with $a = d \cdot i/2$. It follows that Equation (4) can be expressed in the form

$$\frac{x^2}{i^2} - \frac{y^2}{1-i^2} = \frac{d^2}{4}, \tag{5}$$

with asymptotes given by

$$y = \pm\sqrt{\frac{1-i^2}{i^2}} \cdot x. \tag{6}$$

These asymptotes show limiting linear relationships between the coordinates of points (x, y) designated as the target position, that give values $\pm$ i for the fratricide index. We note the expressions in Equations (5) and (6) are valid for $i^2 < 1$ and $i \neq 0$. If $i = 0$, the hyperbola degenerates to the vertical line $x = 0$. If $i = 1$, the hyperbola degenerates to the half-line consisting of the points on or to the right of the focus (c,0), and similarly for $i = -1$.

We used the fratricide index to rate each detection that occurred during a company STX. Figure 13 is an example of the distribution of the fratricide index for a particular mission. There were eight detections in this particular mission. As Figure 13 reveals, only one detection resulted in a negative value.

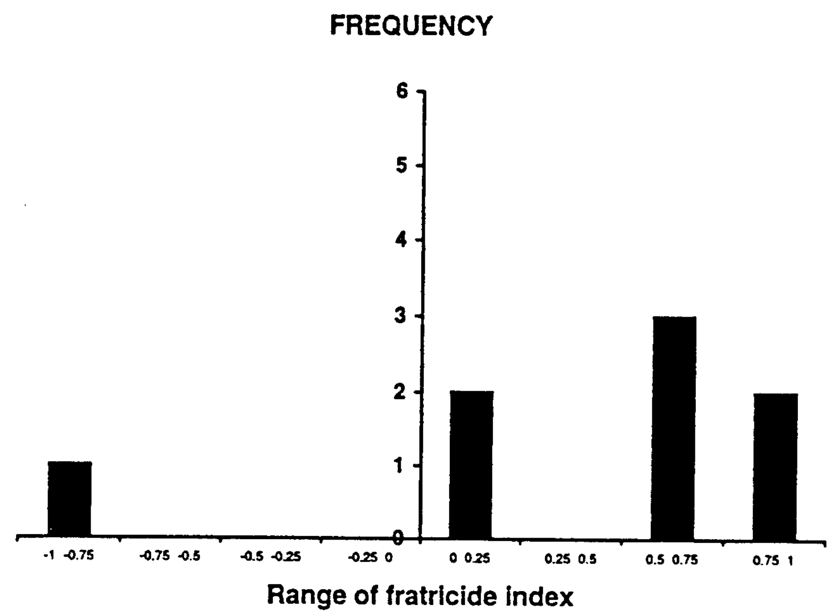



Figure 13. Distribution of detections rated by fratricide index.

## OBSERVATIONS

Appendix A contains summary output generated for each STX mission. The baseline STX trials are shown first followed by the results from treatment STX trials. Since the trials were not conducted in accordance with a complete experimental design, we were unable to make strong statistical comparison of the results. Some of the most notable confounding factors present in the data are unit, scenario, mission, and terrain. The units participating in the treatment STX were not the same units that participated in the baseline STX. The scenarios and missions of the two sets of STX were not the same. Baseline STX were focused on deliberate attack while treatment STX were focused on breaching operations. The trials were conducted in different terrain. Finally, the sample size (eight baseline trials and eight treatment trials) was too small to employ the power of statistical analysis of variance.

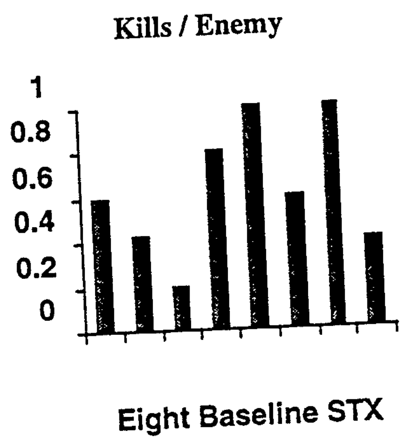
We did, however, observe some notable differences in the output from the two data sets that merit comment. The data supporting our observations are recorded in Table3.

| | % ENEMY KILLED | %ENEMY DETECTED | MEAN FRATRICIDE INDEX | MEAN MAXIMUM INFO GAIN |
|---|---|---|---|---|
| BASELINE | 63% | 142% | .3075 | .6650 |
| TREATMENT | 21% | 74% | .2918 | .3113 |

Table 3. Summary results.

## LEATHALITY

Baseline trials recorded substantially more kills than treatment trials (Figure 14). We report this result in Table 3 as a percentage of kills relative to the size of the enemy. Enemy size varied within the trials from four to seven enemy vehicles. Likewise, the baseline trials generated almost twice the detections of the treatment trials (Figure 15). These results suggest that the treatment actually decreased the lethality of the force. However, these outcomes could easily be due to the confounding effect of the scenario, mission, and terrain factors.



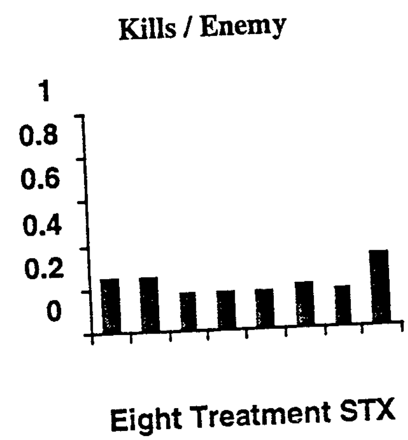




Figure 14. Number of kills per number of enemy present.

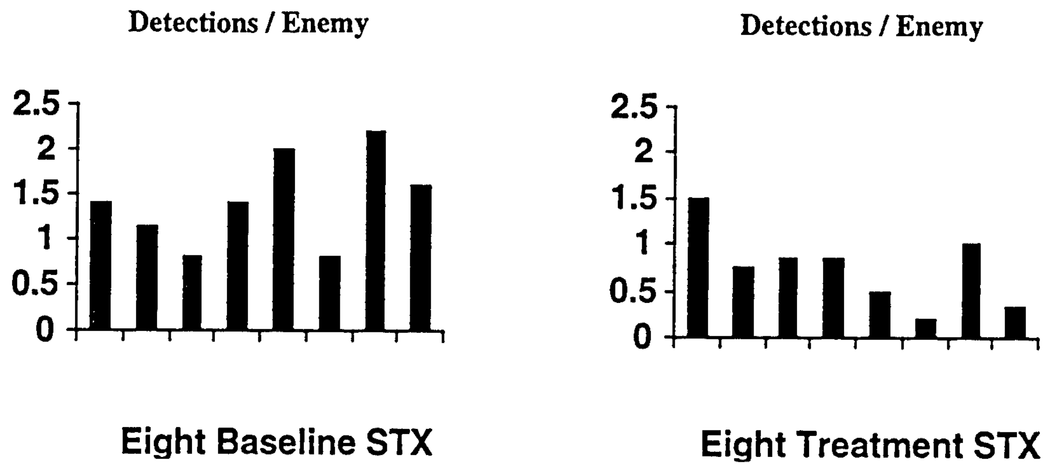



Figure 15. Number of detections per number of enemy present.

## SITUATIONAL AWARENESS

We investigated the cumulative information gain curves in Appendix A to determine the maximum cumulative gain achieved by the Blue force during each mission. As Table 3 reveals, the baseline trials produced the most favorable information gain; the mean value over all baseline STX is greater than twice that of treatment STX. Figure 16 shows the distribution of the maximum values achieve during all missions. Baseline units seemed to learn more about the enemy than treatment units. Again, the results could be attributed to the scenario, mission, and terrain factors. Also, we may be seeing a marginal return on information gain; units in baseline trials may have had more to learn than their counterparts in treatment trials. In other words we assumed an uninformed unit crossed the line of departure (LD) at the beginning of the battle. The technology available to units in treatment trials may have provided these units with significant information about the enemy's disposition prior to LD. This awareness on the part of Blue, if it existed, is not captured in our displays of information gain. If "stronger priors" representing Blue's more informed state about Red's disposition had been used, however, the information gain for the treatment STX trials would have been lower than those we report. On the other hand, the maximum entropy used to normalize information gain would also have been smaller. The net effect on values of maximum information gain shown in Figure 16 is difficult to assess.

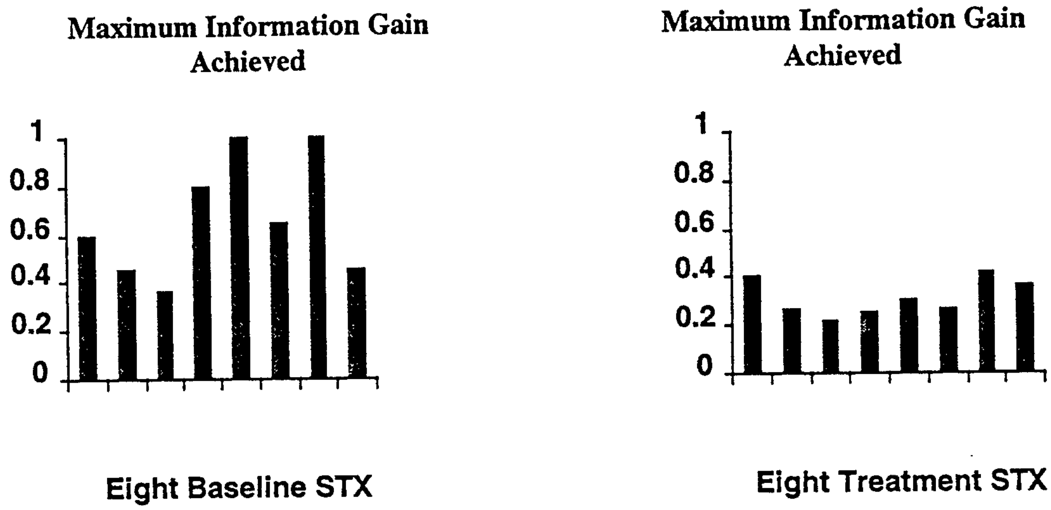



Figure 16. The maximum value of information gain in each trial.

## FRATRICIDE

Mean values of the fratricide index suggest no difference in potential fratricide. As with information and lethality measures, confounding factors are present in these results. However, due to the relative nature of the fratricide index computations we do not believe it to be sensitive to the factors of mission and scenario. Terrain, however, is clearly a factor which could have influenced these outcomes.

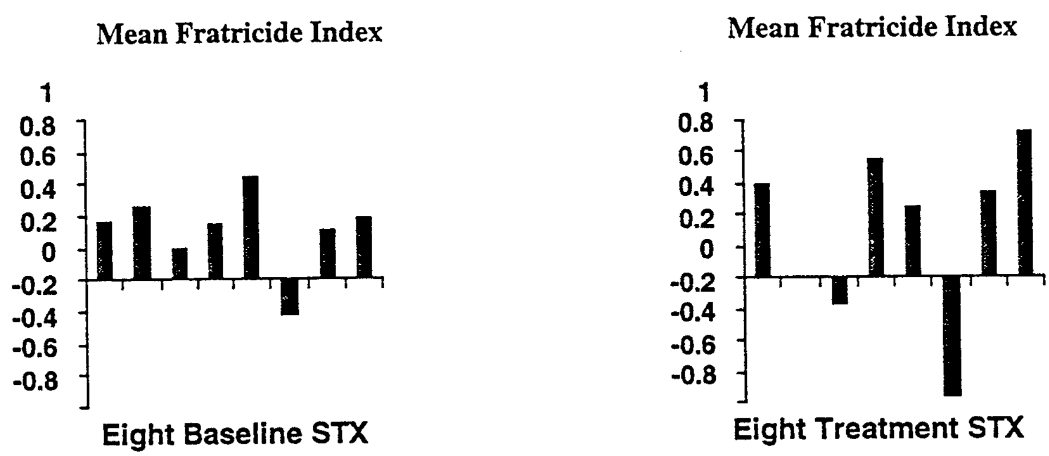



Figure 17. Average fratricide index over all detections within each trial.

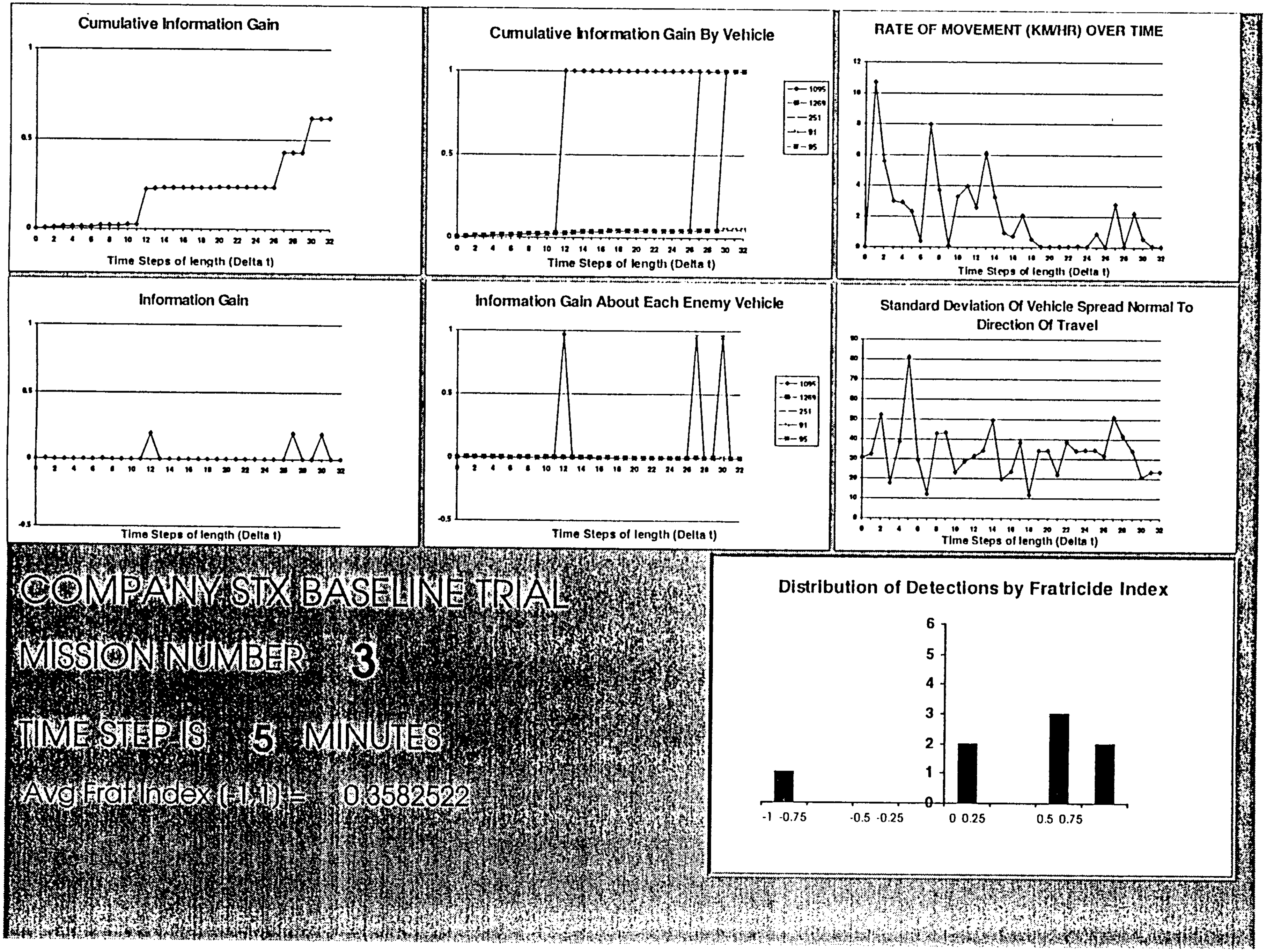
## CONCLUSIONS

We have demonstrated that the information gain measure can be computed from digitally recorded position data such as those generated during the AWE. We see evidence the measure behaves rationally, i.e. units that were effective at finding and killing the enemy had better information gain curves than units that did not. Likewise, units that were able to kill the enemy were rewarded over those that could only detect. This is primarily due to the information degradation process in our model.

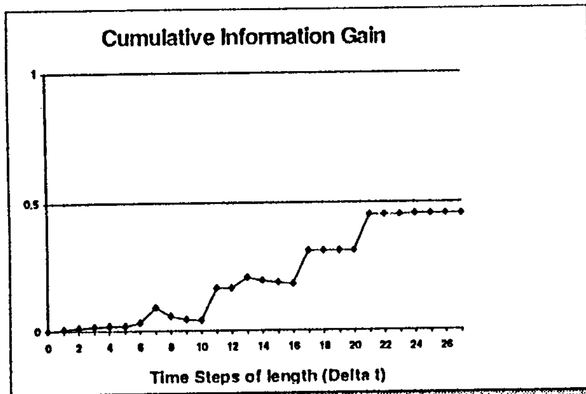
Our approach to evaluating detections gave rise to an apparently new measure of fratricide which we have documented here for the first time. This measure may become more applicable to analysts and warfighters as units begin to acquire digital position tracking equipment. It is conceivable that units will perform analysis of their training data using the ground truth that digital location information provides. Since the fratricide index is easy to compute and understand it could provide training units with relevant feedback concerning their detection and identification of enemy forces.

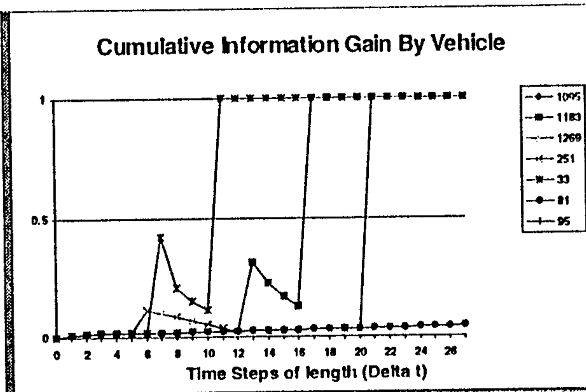
We were limited in making strong comparisons between baseline and treatment results because of uncontrolled factors in the trials. The same units did not complete both baseline and treatment trials. The companies of one brigade conducted the baseline trials and the companies of another the treatment trials. Each brigade prepared their own training plans and scenarios for STX; one brigade's companies did a deliberate attack, the other a breach mission. Additionally, the two training events were executed on different terrain during different seasonal conditions. We were able to comment on differences but are not convinced that the differences are attributable to the treatments.

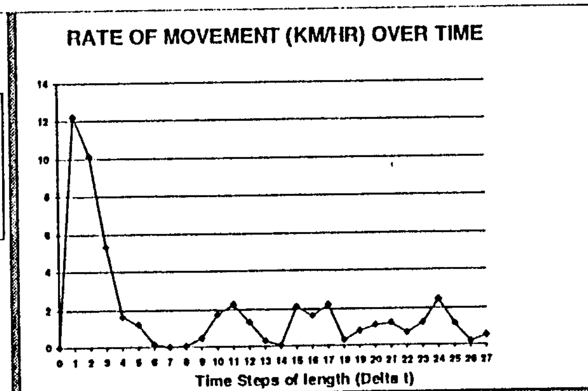
# Appendix A. Summary output for baseline STX trials.

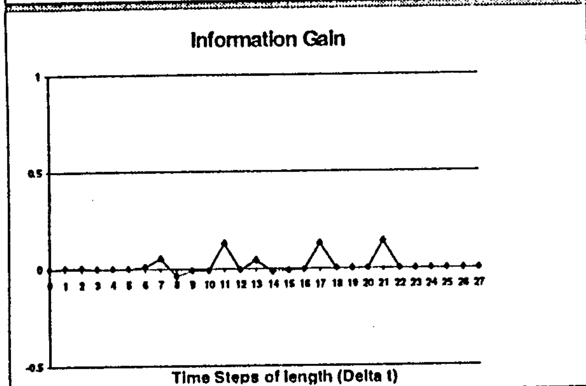


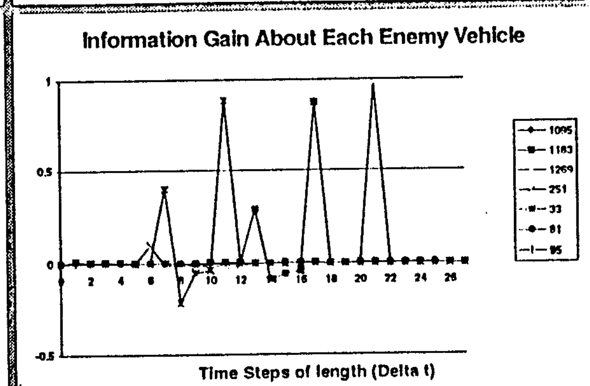
Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1095
1269
251
91
95
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1095
1269
251
91
95
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX BASELINE TRIAL
MISSION NUMBER 3
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.3582522
Distribution of Detections by Fratricide Index
-1 -0.75
-0.5 -0.25
0 0.25
0.5 0.75

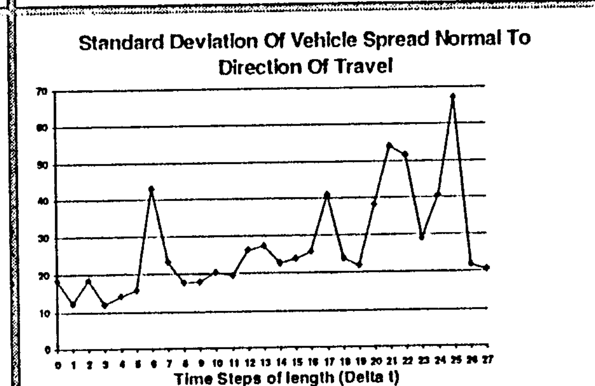


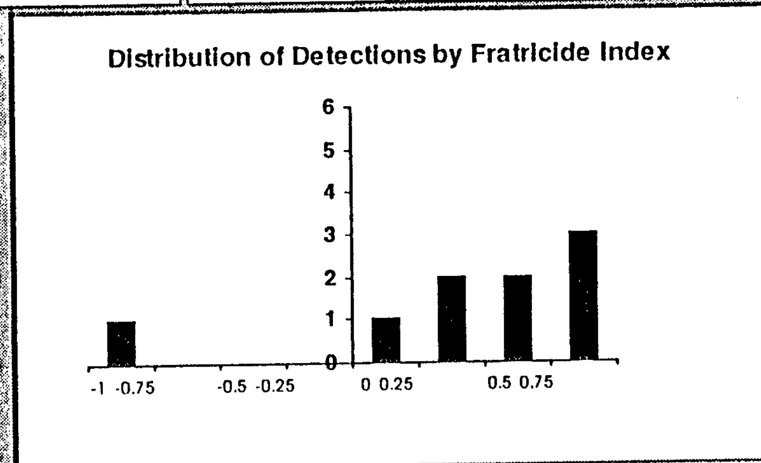



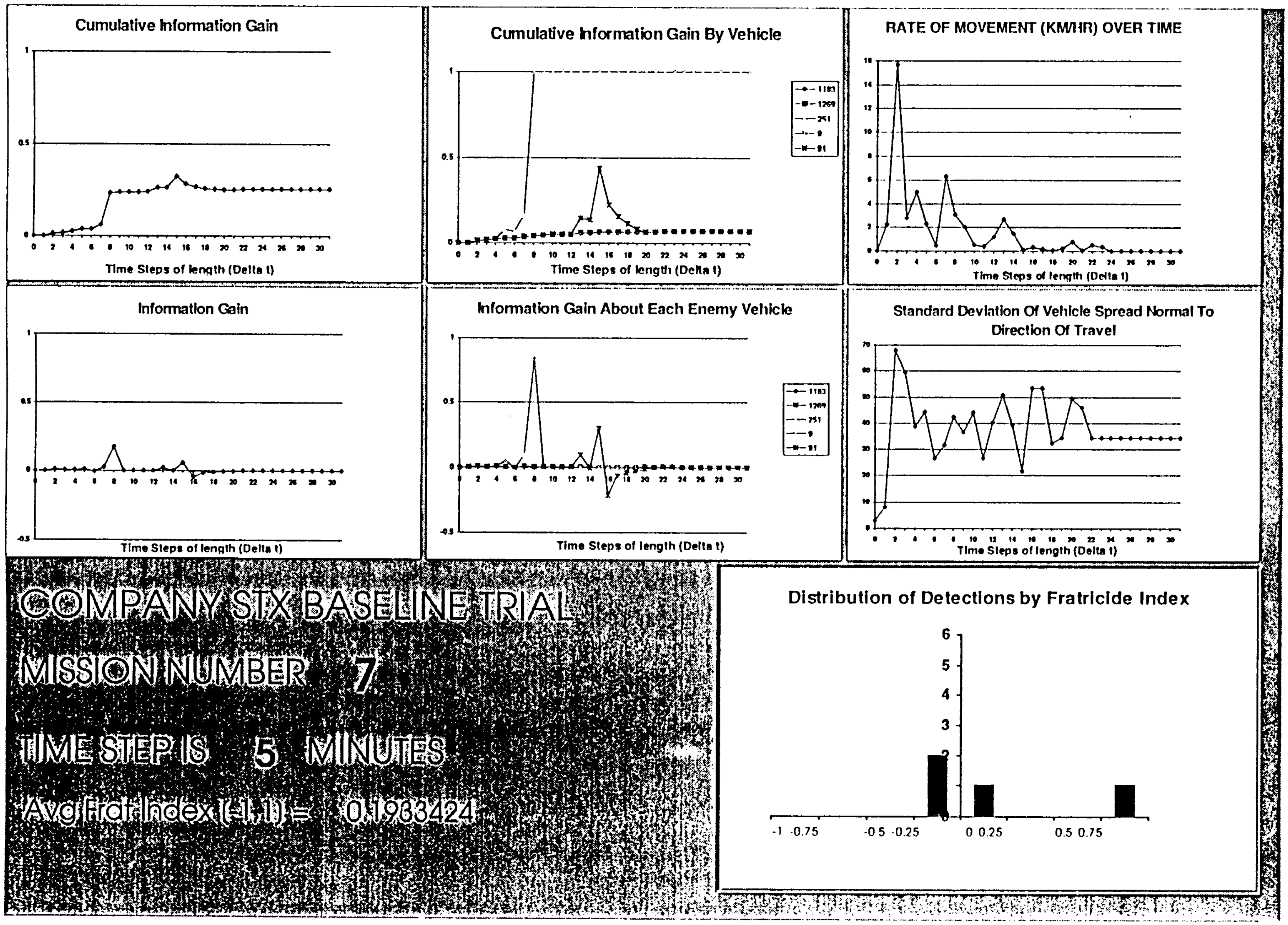

COMPANY STX BASELINE TRIAL

MISSION NUMBER 6

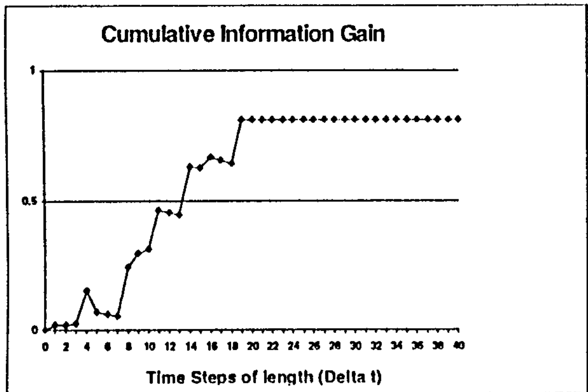
TIME STEP IS 5 MINUTES

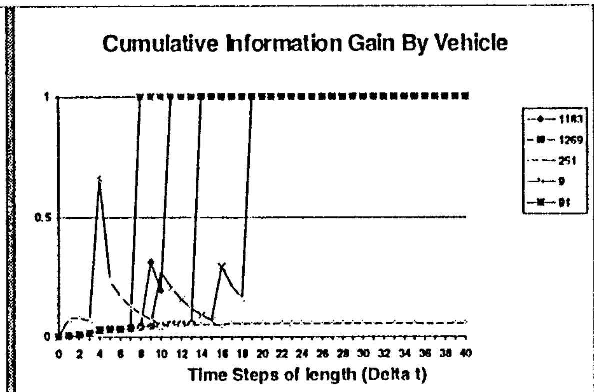
Avg Frat Index (-1,1) = 0.4606721

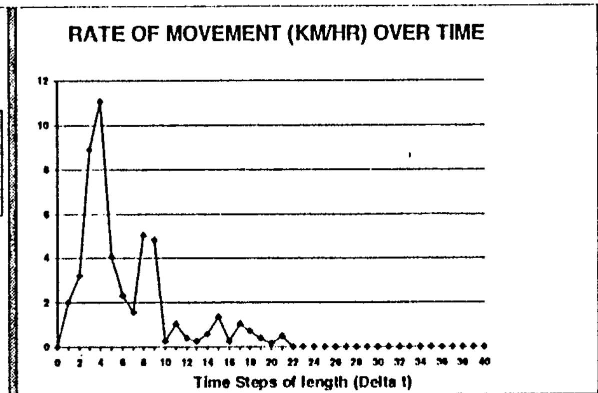


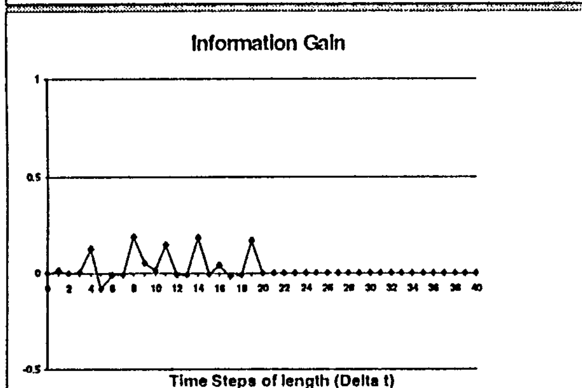


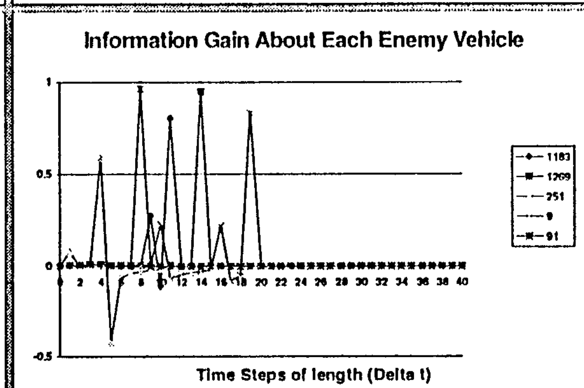
Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1183
1269
251
9
91
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1183
1269
251
9
91
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX BASELINE TRIAL
MISSION NUMBER 7
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.1983424
Distribution of Detections by Fratricide Index

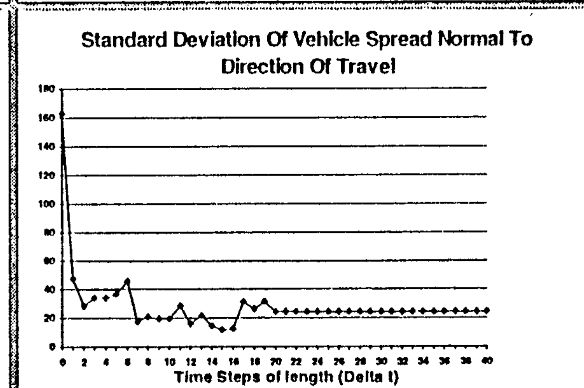


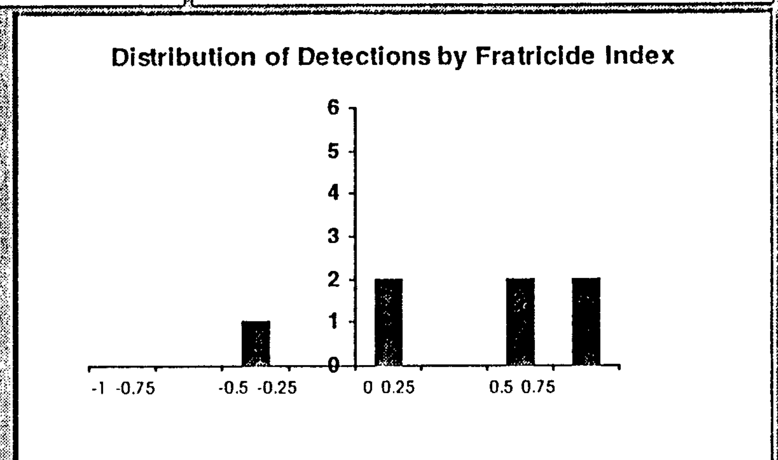



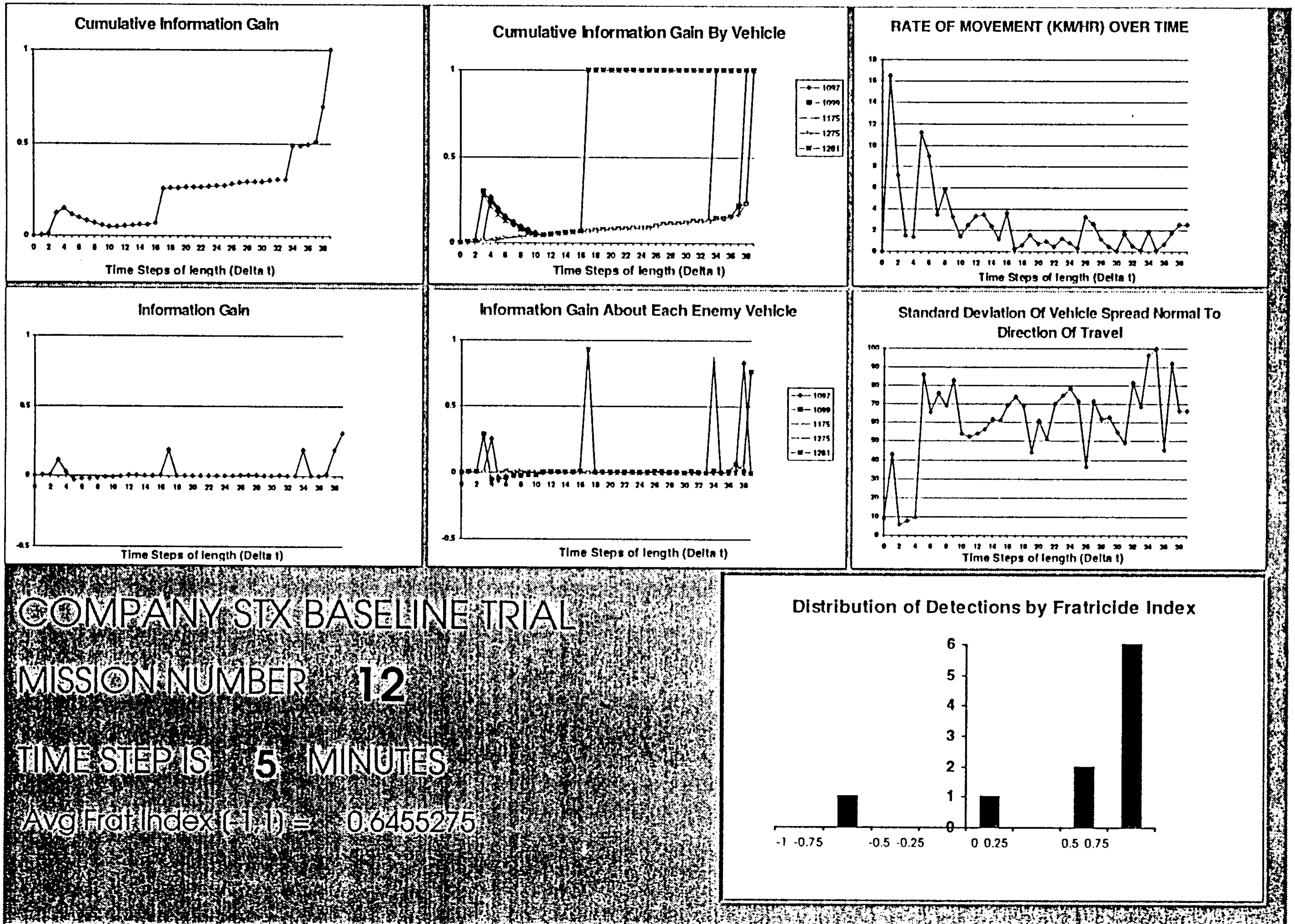

COMPANY STX BASELINE TRIAL

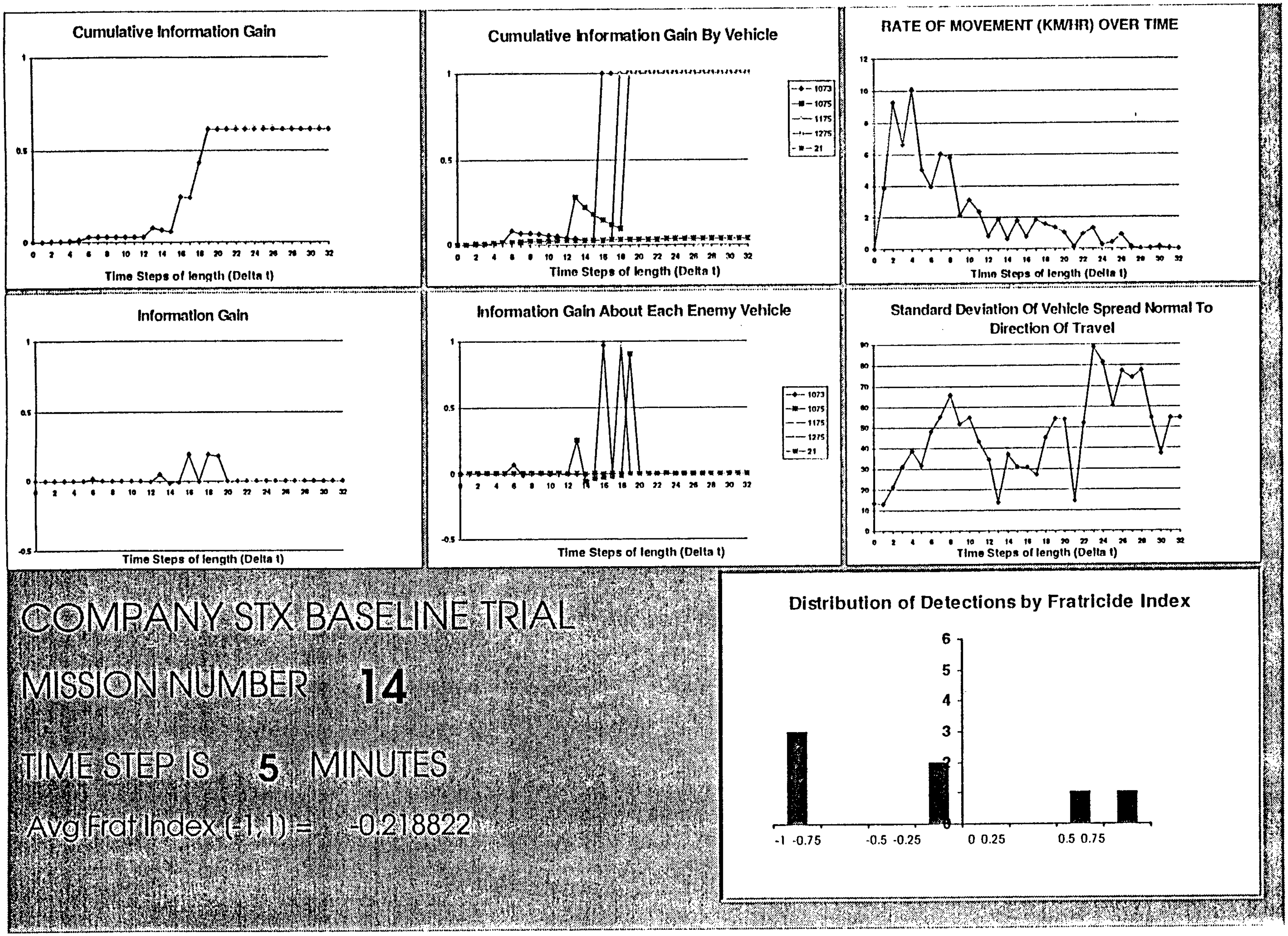
MISSION NUMBER 8

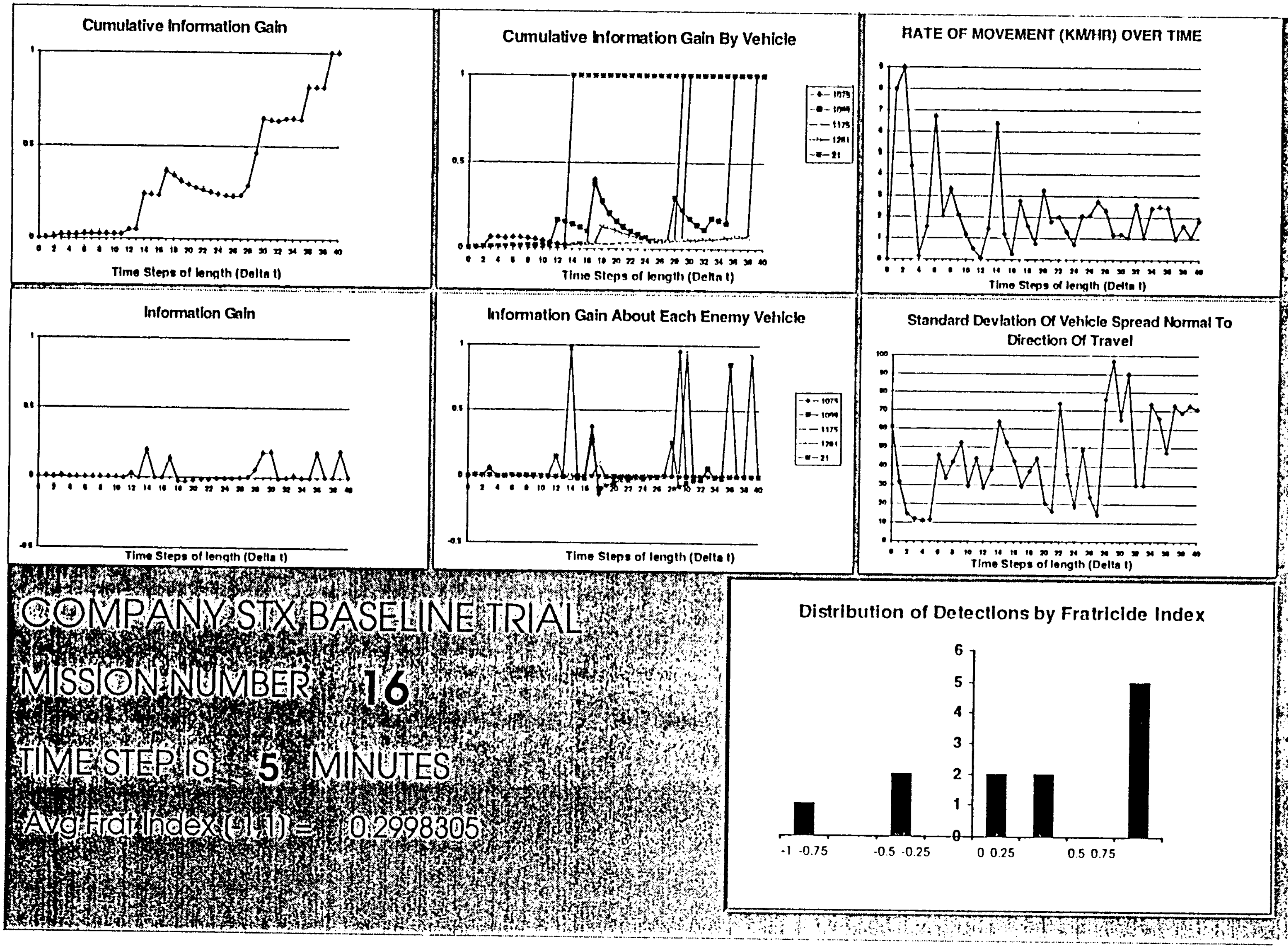
TIME STEP IS 5 MINUTES

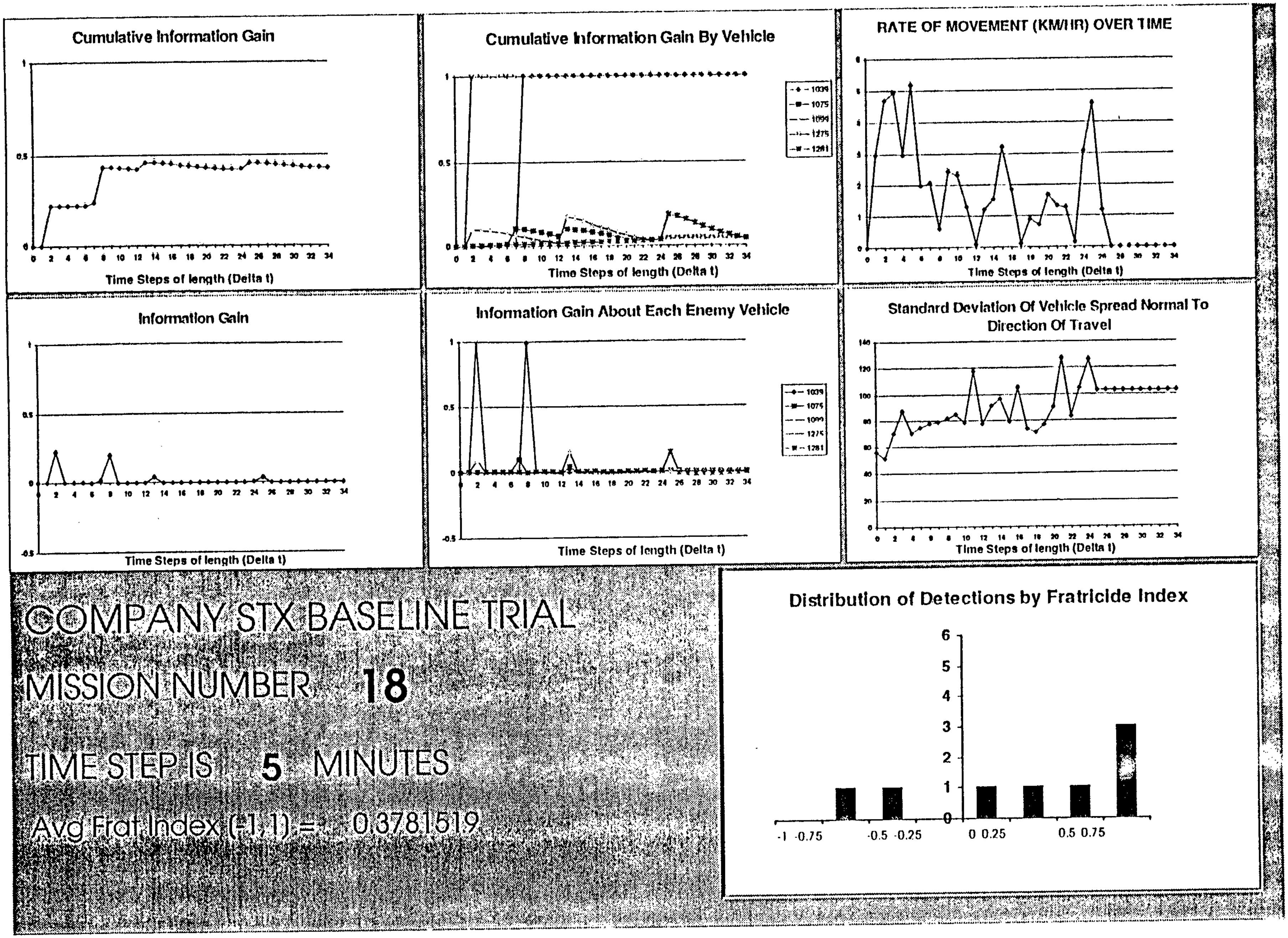
Avg Frat Index (-1,1) = 0.3434176



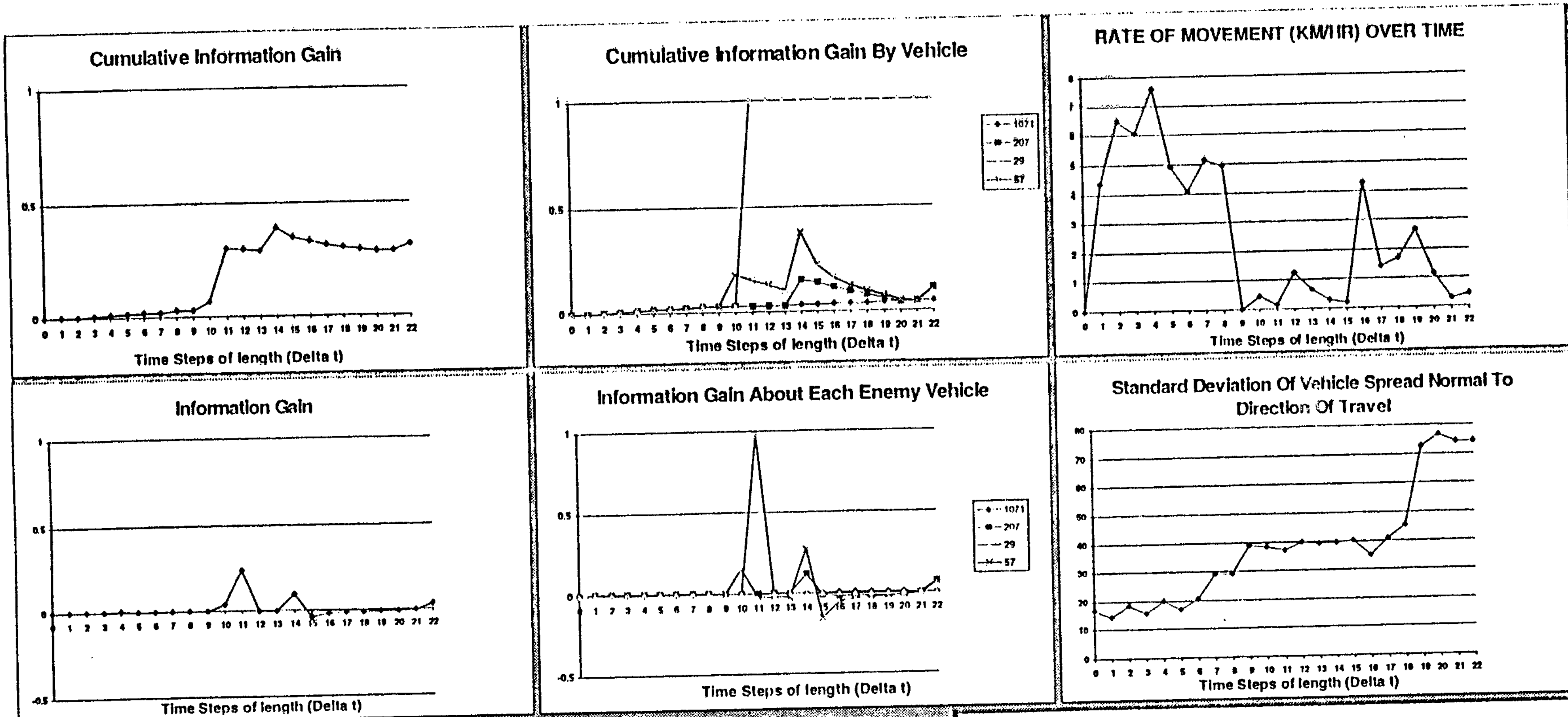

Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1097
1099
1175
1275
1281
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1097
1099
1175
1275
1281
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX BASELINE TRIAL
MISSION NUMBER 12
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.6455275
Distribution of Detections by Fratricide Index
6
5
4
3
2
1
0
-1 -0.75
-0.5 -0.25
0 0.25
0.5 0.75


Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1073
1075
1175
1275
21
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1073
1075
1175
1275
21
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX BASELINE TRIAL
MISSION NUMBER 14
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = -0.218822
Distribution of Detections by Fratricide Index
-1 -0.75
-0.5 -0.25
0 0.25
0.5 0.75


Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1075
1099
1175
1261
21
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1075
1099
1175
1261
21
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX BASELINE TRIAL
MISSION NUMBER 16
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.2998305
Distribution of Detections by Fratricide Index
-1 -0.75
-0.5 -0.25
0 0.25
0.5 0.75


Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX BASELINE TRIAL
MISSION NUMBER 18
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.3781519
Distribution of Detections by Fratricide Index
-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75

## Appendix B. Summary output for treatment STX trials.

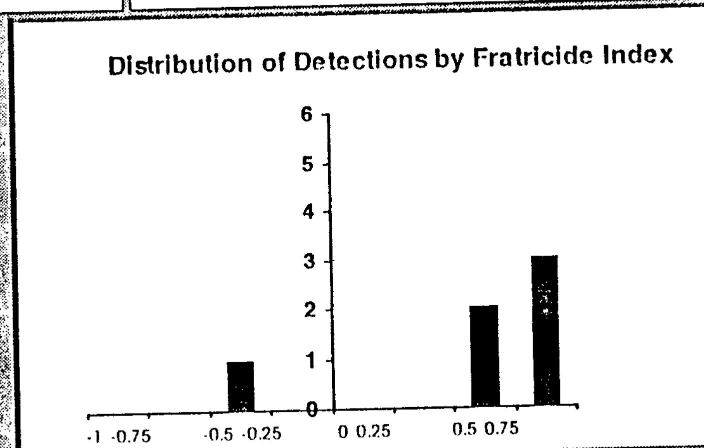


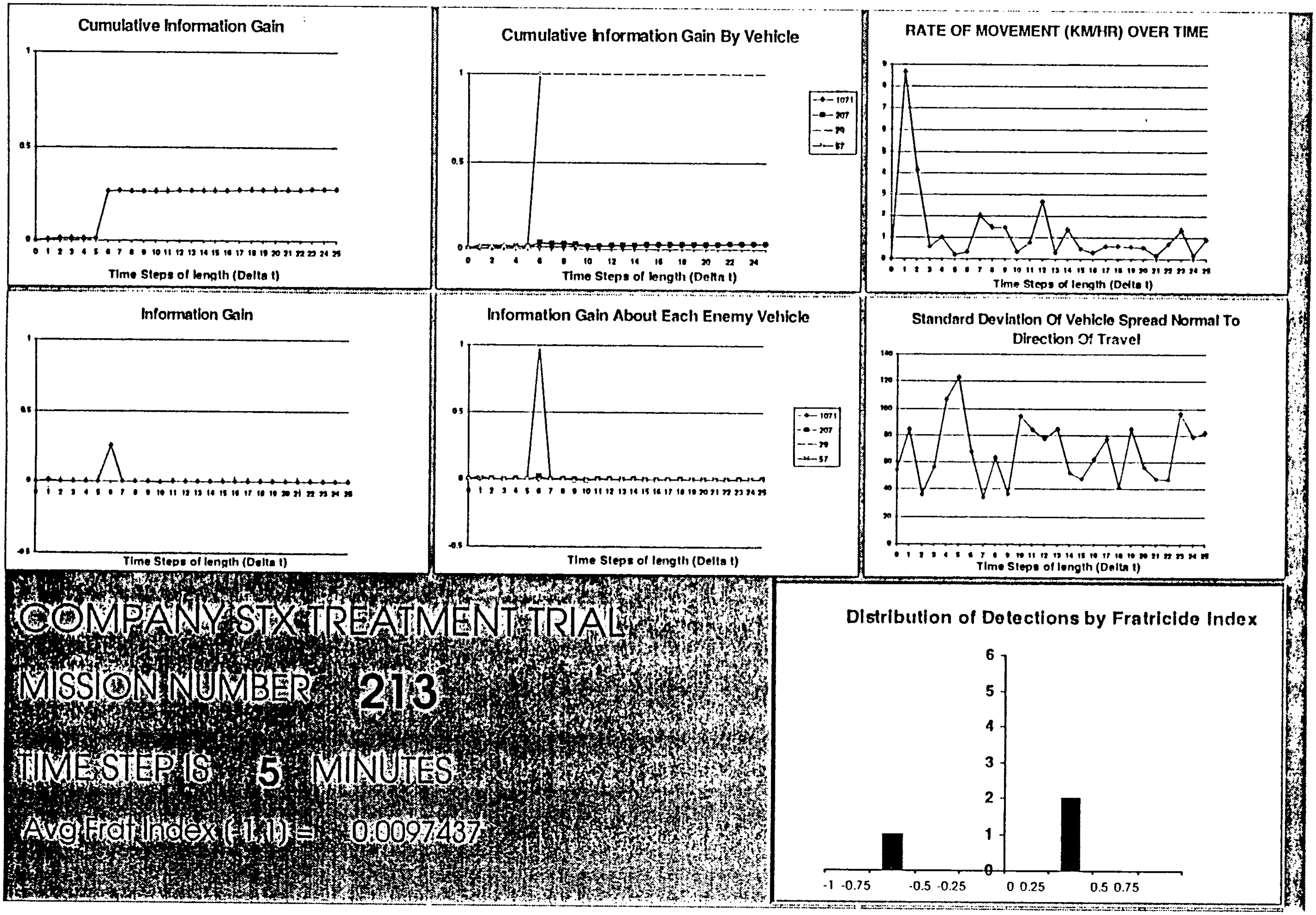

COMPANY STX TREATMENT TRIAL

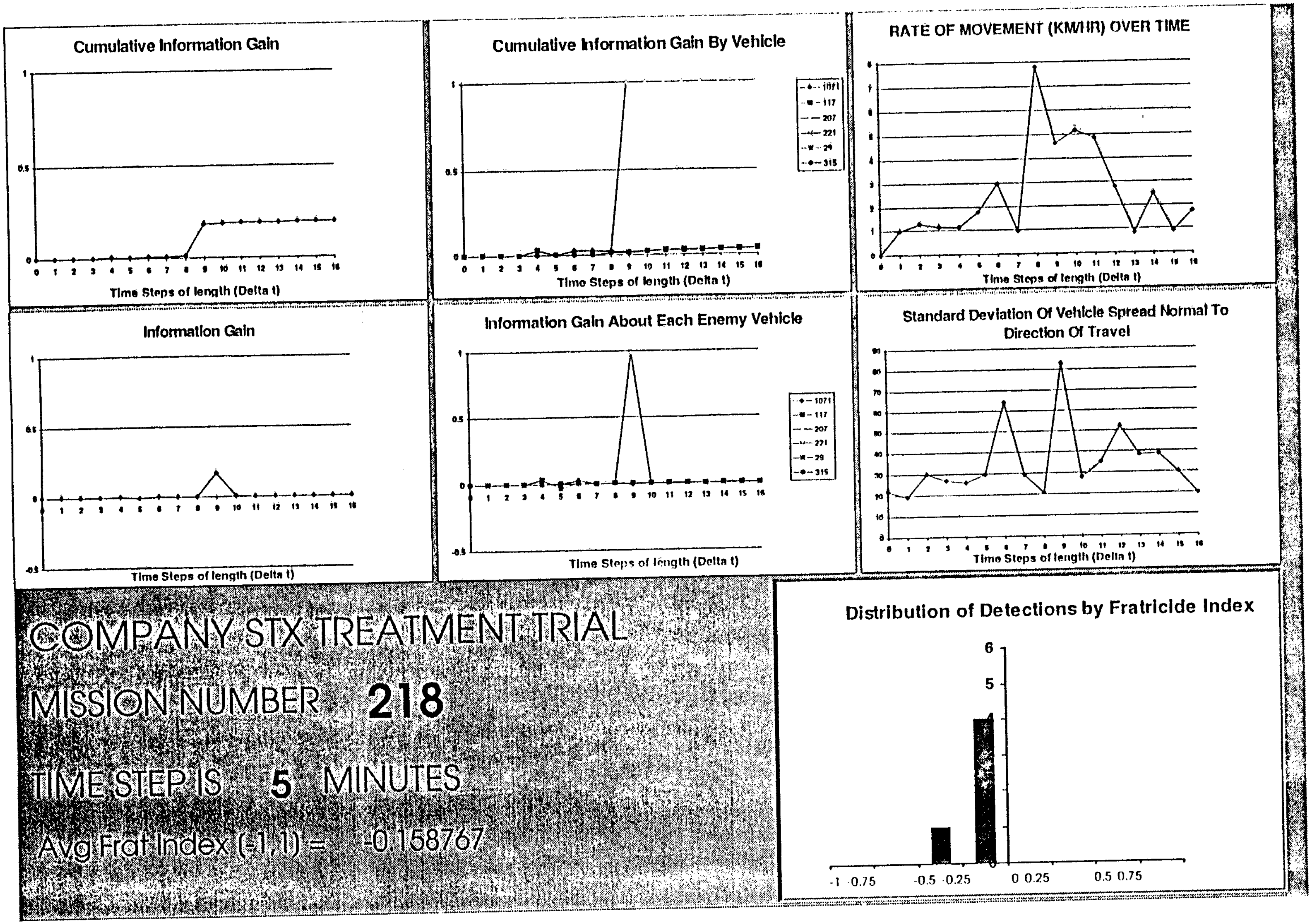
MISSION NUMBER **209**

TIME STEP IS **5** MINUTES

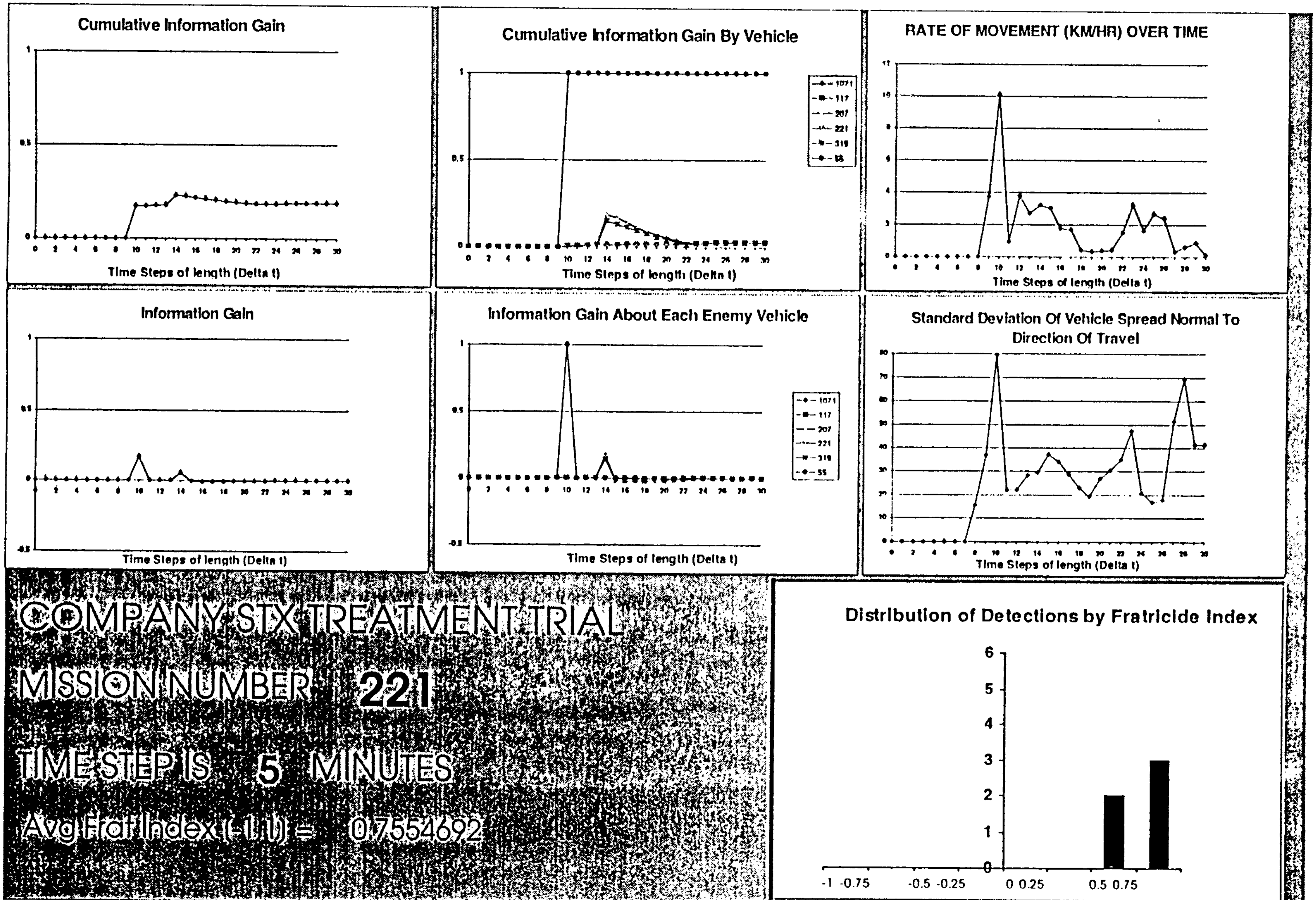
Avg Frat Index (-1,1) = 0.6055301

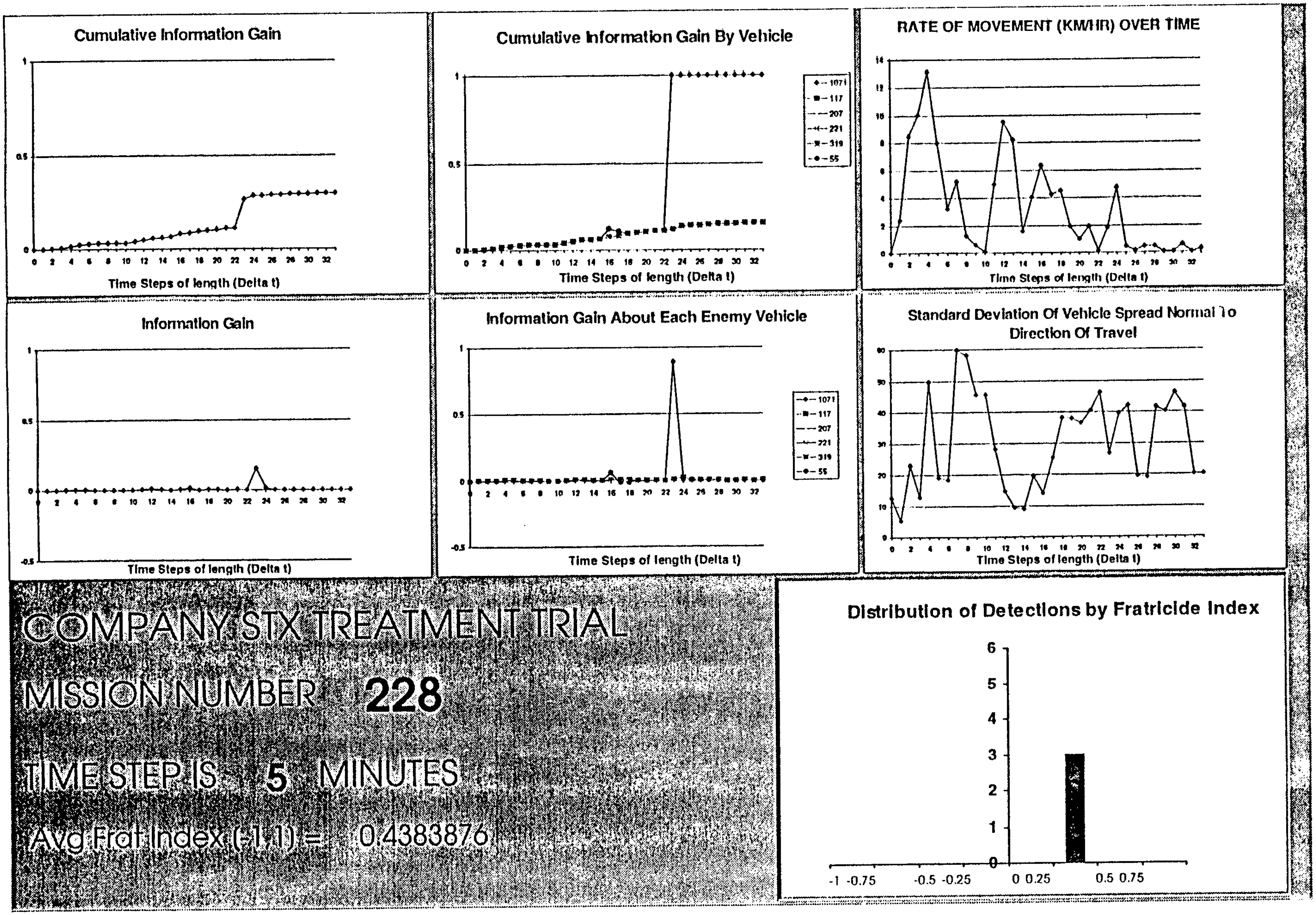


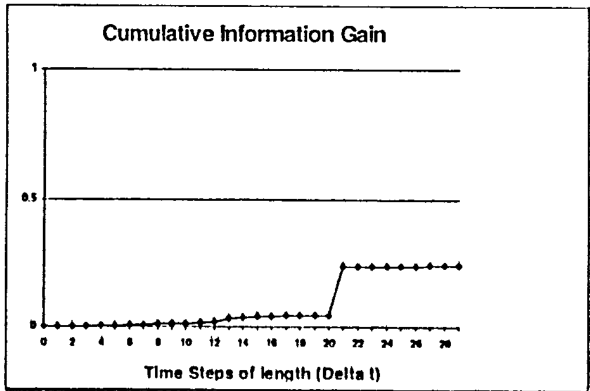

Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1071
207
79
57
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1071
207
79
57
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX TREATMENT TRIAL
MISSION NUMBER 213
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.0097437
Distribution of Detections by Fratricide Index
6
5
4
3
2
1
0
-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75

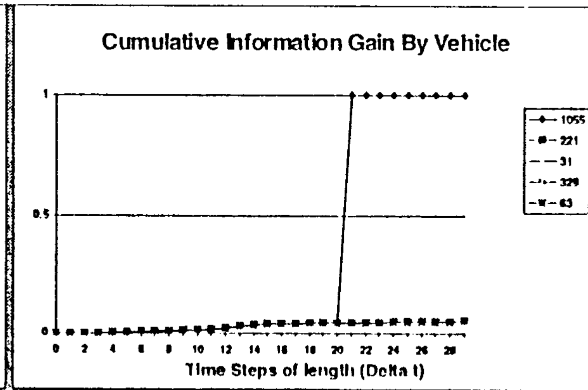


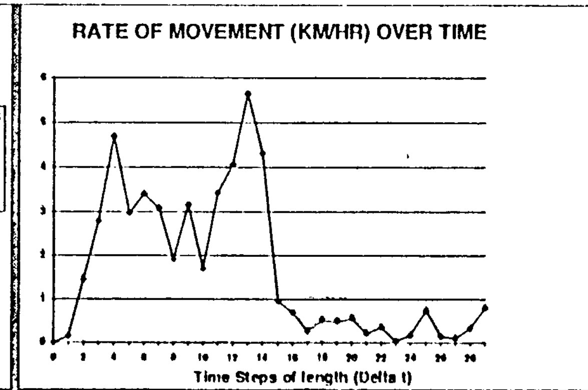
Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1071
117
207
221
29
315
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX TREATMENT TRIAL
MISSION NUMBER 218
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = -0.158767
Distribution of Detections by Fratricide Index
-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75

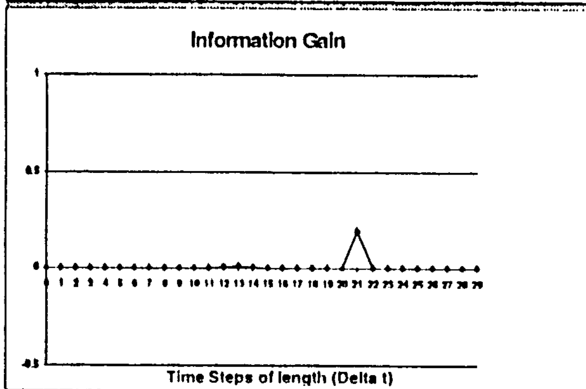

Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1071
117
207
221
319
55
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1071
117
207
221
319
55
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX TREATMENT TRIAL
MISSION NUMBER 221
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.7554692
Distribution of Detections by Fratricide Index
-1 -0.75
-0.5 -0.25
0 0.25
0.5 0.75

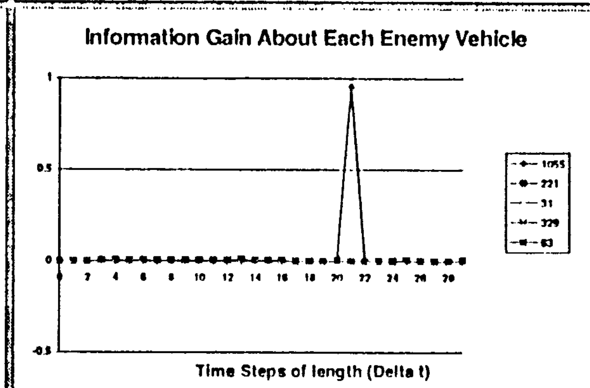

Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1071
117
207
221
319
55
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1071
117
207
221
319
55
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX TREATMENT TRIAL
MISSION NUMBER 228
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.4383876
Distribution of Detections by Fratricide Index
6
5
4
3
2
1
0
-1 -0.75
-0.5 -0.25
0 0.25
0.5 0.75

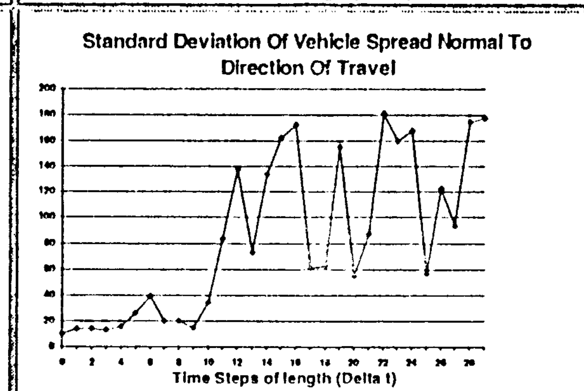


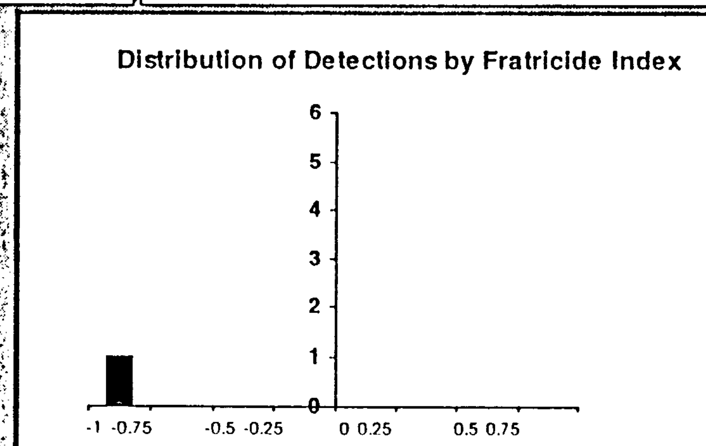



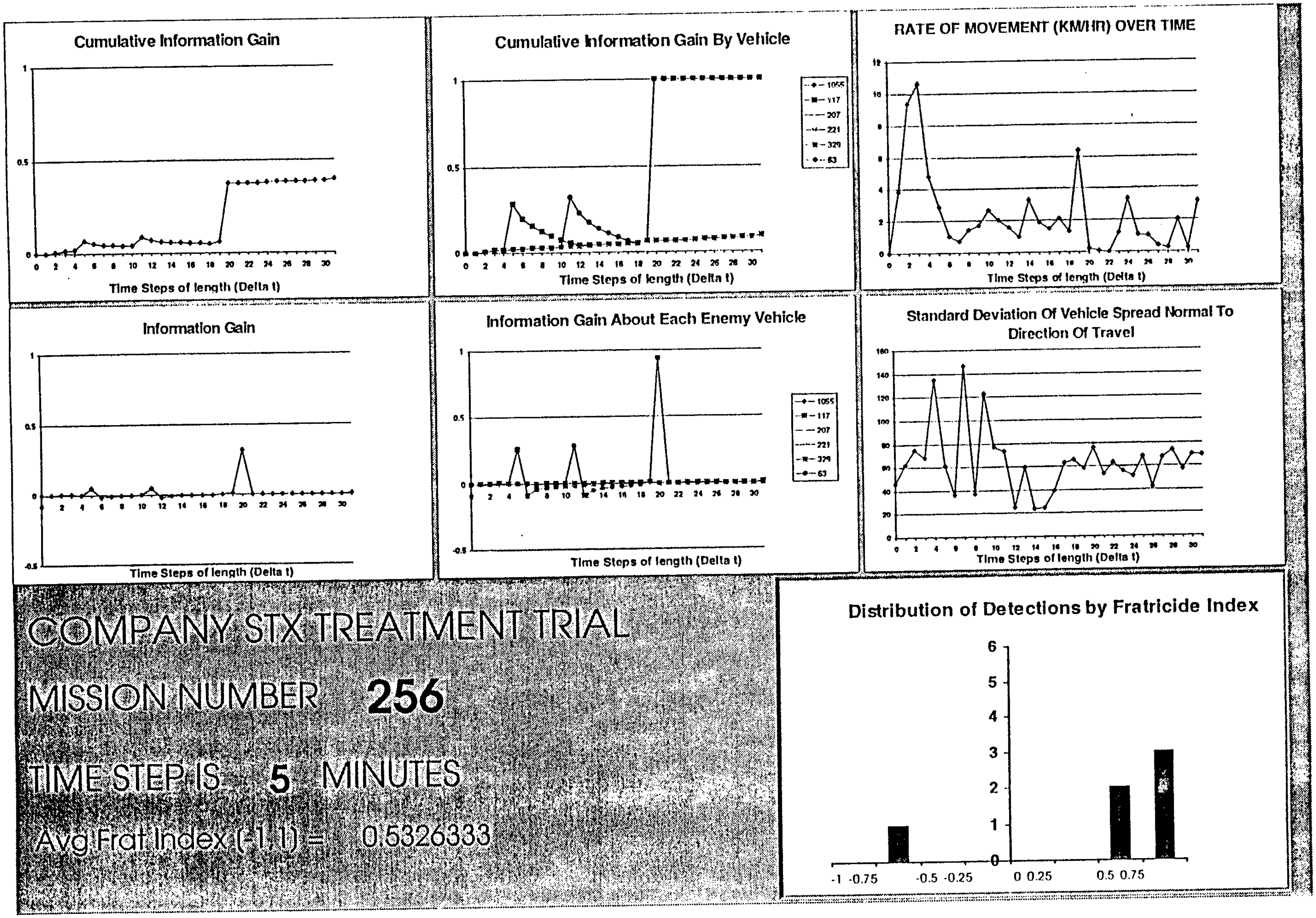

COMPANY STX TREATMENT TRIAL

MISSION NUMBER 240

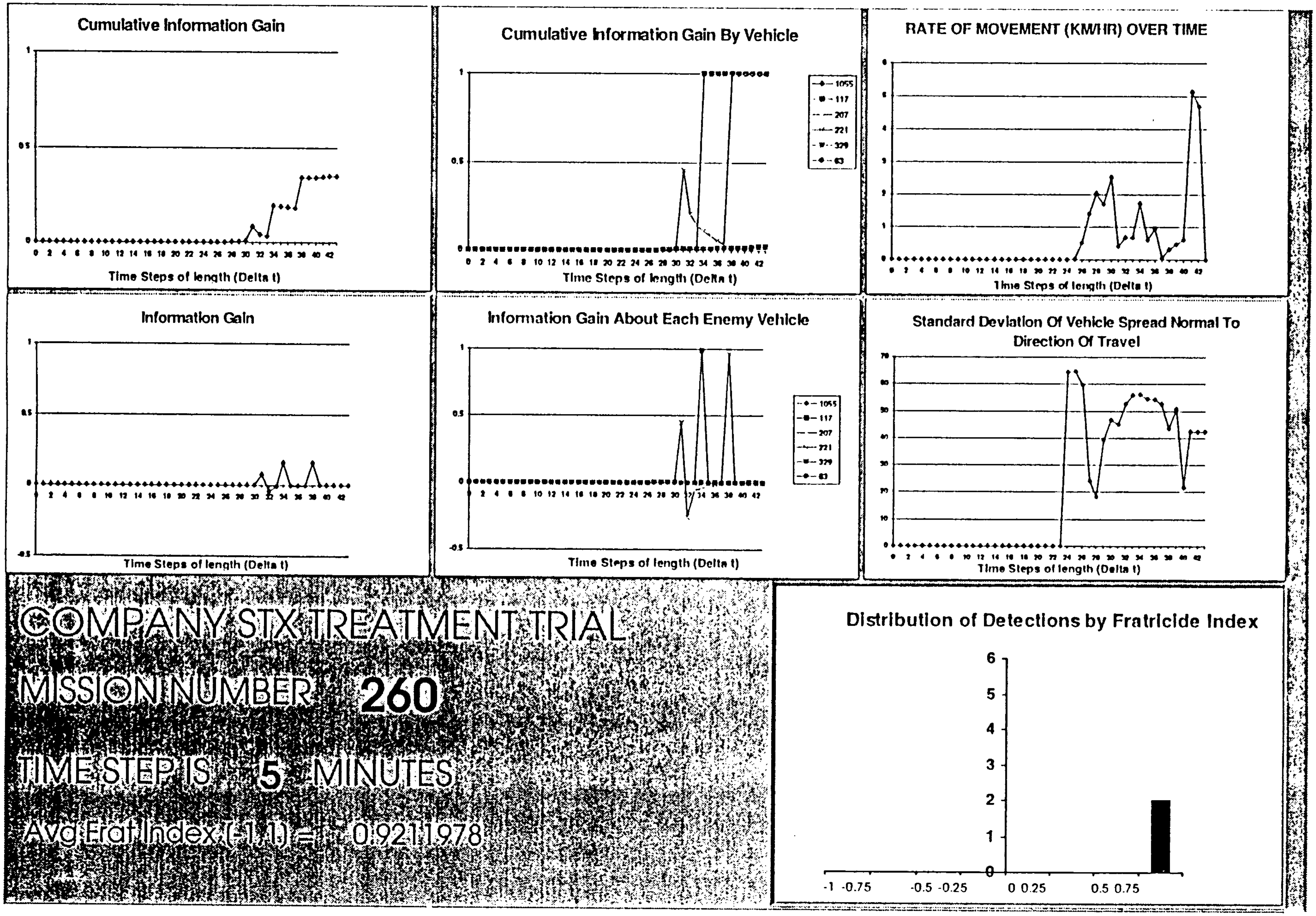
TIME STEP IS 5 MINUTES

Avg Frdt Index (-1,1) = -0.769237





Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1055
117
207
221
329
63
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1055
117
207
221
329
63
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX TREATMENT TRIAL
MISSION NUMBER 256
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.5326333
Distribution of Detections by Fratricide Index
-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75


Cumulative Information Gain
Time Steps of length (Delta t)
Cumulative Information Gain By Vehicle
1055
117
207
221
329
83
Time Steps of length (Delta t)
RATE OF MOVEMENT (KM/HR) OVER TIME
Time Steps of length (Delta t)
Information Gain
Time Steps of length (Delta t)
Information Gain About Each Enemy Vehicle
1055
117
207
221
329
83
Time Steps of length (Delta t)
Standard Deviation Of Vehicle Spread Normal To Direction Of Travel
Time Steps of length (Delta t)
COMPANY STX TREATMENT TRIAL
MISSION NUMBER 260
TIME STEP IS 5 MINUTES
Avg Frat Index (-1,1) = 0.9211978
Distribution of Detections by Fratricide Index
-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75

## APPENDIX C: Documentation and explanation of code.

RAW DATA

OPTEC provided raw data in five tabulated files. The scenario file (DBSCENAR) contains vehicle type and identification for Red and Blue players as shown in Table 4.

| IDX | MISSNUM | DTG | PID | PLAYER | FORCE | PLATFORM |
|---|---|---|---|---|---|---|
| 10034 | 011 | 960408234716 | 13 | D13 1/D/1-67 AR | BL | M1A1 |
| 10039 | 011 | 960408234716 | 25 | D24 2/D/3-67 AR | BL | M1A1 |
| 10042 | 011 | 960408234716 | 33 | D34 3/D/1-67 AR | OP | T80 |
| 10044 | 011 | 960408234716 | 41 | D22 2/D/3-67 AR | BL | M1A1 |
| 10050 | 011 | 960408234716 | 57 | HQ137 1/C/1-67 AR | OP | BMP |
| 10058 | 011 | 960408234716 | 81 | D66 1/A/1-67 AR | OP | T80 |
| 10059 | 011 | 960408234716 | 83 | D12 1/D/3-67 AR | BL | M1A1 |
| 10061 | 011 | 960408234716 | 89 | D23 2/D/3-67 AR | BL | M1A1 |
| 10064 | 011 | 960408234716 | 99 | D66 1/D/3-67 AR | BL | M1A1 |

Table 4. Sample of scenario data provided by client.

The location file (DBLOCATN) contains the positioning information of all vehicles during the battle. See Table 5.

| SITE | CELL | MISSNUM | DATE | PID | PLAYER | S_TIME | E_TIME | POSITION |
|---|---|---|---|---|---|---|---|---|
| HX | C | 004 | 960326 | 101 | B31 3/B/3-67 AR | 235954 | 000012 | PK14527286 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001608 | 001623 | PK14537288 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001623 | 001657 | PK14527285 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001657 | 001715 | PK14537288 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001715 | 001811 | PK14527285 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001811 | 001826 | PK14537288 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001826 | 001900 | PK14537286 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 001900 | 002013 | PK14537288 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 002013 | 002027 | PK14527288 |
| HX | C | 004 | 960327 | 101 | B31 3/B/3-67 AR | 002027 | 002048 | PK14527289 |

Table 5. Sample of position location data provided by client

The detections file contains the time and location of Blue detections of Red vehicles. See Table 6.

| ID | MSN | TIME | LOCATION | ENEMY |
|---|---|---|---|---|
| 1 | 3 | 26MAR96:10:30 | PK138668 | T80 |
| 2 | 3 | 26MAR96:10:30 | PK182702 | BMP |
| 3 | 3 | 26MAR96:10:30 | PK188708 | BMP |
| 4 | 3 | 26MAR96:10:30 | PK156709 | BMP |
| 5 | 3 | 26MAR96:10:30 | PK171699 | T80 |
| 6 | 3 | 26MAR96:10:30 | PK168701 | T80 |
| 7 | 3 | 26MAR96:10:30 | PK176703 | T80 |

Table 6. Sample of detections provided by client.

The mission data is a simple tabulation of mission start times and end times. See Table 7.

| MSN | START | END |
|---|---|---|
| 3 | 26MAR96:07:48 | 26MAR96:10:30 |
| 6 | 28MAR96:08:17 | 28MAR96:10:33 |
| 7 | 29MAR96:07:54 | 29MAR96:10:30 |
| 8 | 30MAR96:07:08 | 30MAR96:10:30 |
| 9 | 02APR96:07:43 | 02APR96:10:35 |
| 10 | 04APR96:07:14 | 04APR96:10:30 |
| 12 | 09APR96:07:43 | 09APR96:11:00 |

Table 7. Sample of mission data provided by client.

Finally, the kill table contains the time of death and identification of Blue kills of Red vehicles. See Table 8.

| MSN | PID | KTime | BUMPERNO |
|---|---|---|---|
| 008 | 1183 | 7:58:00 AM | A32 |
| 008 | 1269 | 8:14:00 AM | A21 |
| 008 | 9 | 8:41:00 AM | A31 |
| 008 | 91 | 7:44:00 AM | A65 |

Table 8. Sample kill data provided by client

## DATA REDUCTION TOOL

As mentioned above we developed the data reduction and analysis tool in Microsoft Access. The tool is driven by the user selection of a mission. Essentially, the user selects a mission and the analysis tool does all the rest; it converts raw data to usable formats, calculates the information gain measure, the fratricide index, etc. and makes various reports and forms available for display and printing.

We do not discuss the user interface in this report since it is essentially a product of standard Access routines. The bulk of this appendix documents the code behind the interface. We show how the code accesses the raw data and produces the desired products. We show the format of the output tables but do not display the reports, charts, etc. which are essentially products of these output tables and can be easily reproduced using the output data.

Comments in the code are shown in boldface type. In most instances queries are developed in the code using SQL. These are relatively easy to interpret by looking closely at the code. In some cases however we reference queries which we developed in the Access user interface. These queries represent minor manipulations of the data from the raw data table and we do not document their development.

## DATA TABLE INITIALIZATION

We initially transform the raw data tables provided by OPTEC into data tables whose content and structure support computation of the measures. We require an enemy kill table which list the mission number, enemy vehicle identification, time of death, and bumper number. See Table 9 and the code that follows.

| MSN | PID | KTime | BUMPERNO |
|---|---|---|---|
| 008 | 1183 | 7:58:00 AM | A32 |
| 008 | 1269 | 8:14:00 AM | A21 |
| 008 | 9 | 8:41:00 AM | A31 |
| 008 | 91 | 7:44:00 AM | A65 |

Table 9. Example kill table.

This table is a subset of the kill table shown above in the raw data. The raw data kill table contains kills for all missions. The table developed during initialization reflects only those kills pertaining to the mission selected by the user.

```
'This routine clears the kill table of any data from previous runs and writes the kill
'table to represent the current mission.
'Declarations
Option Compare Database
Option Explicit

Sub BuildKill()
   Dim dbs As DATABASE, tdf As TableDef, pid, MSN, ktme, postn As Field
   Dim strSQL As String, qdf As QueryDef
   Dim MyRecords, MySet As Recordset
   Dim SQLquery As String
   Dim zeroes As String
   Dim miznum

'Return Database object pointing to current database.
 Set dbs = CurrentDb
'Check if Kill_tbl exist
```

```
'If a kill table already exist, then empty it
If isTable("Kill_tbl") Then
   strSQL = "Delete*from Kill_tbl;"
   Set qdf = dbs.CreateQueryDef("", strSQL)
   qdf.Execute
End If
'Retrieve the mission number form the [SPECIFY MISSION FORM] which is up
and running once the user selects "initialize data tables."
miznum = [Forms]![SPECIFY MISSION_frm].[msnnum]
'This is a string fixing routine to put the mission number in the correct string
format.
If Len(miznum) = 2 Then
zeroes = "0"
Else
zeroes = "00"
End If
'Open kill table.
Set MyRecords = dbs.OpenRecordset("Kill_tbl")
'Populate the kill table by retrieving the vehicle ID (PID), the kill time, and bumper
number from an inner join of two queries (DatetoKill_qry and Bumper#). Since
bumper numbers are the same for the enemy on many missions we use this inner
join to select based on a like bumper number and a like date.

'Define the select query.
SQLquery = "SELECT DISTINCT DatetoKill_qry.MSSN, [Bumper#].PID,
striptime(Mid([Time],9)) AS KTime, DatetoKill_qry.BUMPERNO FROM
DatetoKill_qry INNER JOIN [Bumper#] ON (DatetoKill_qry.BUMPERNO =
[Bumper#].BUMP) AND (DatetoKill_qry.Date = [Bumper#].Date) WHERE
(((DatetoKill_qry.MSSN) Like '" & zeroes & miznum & "'))"
'Run the query.
Set MySet = dbs.OpenRecordset(SQLquery)
'Check to see if the query produced zero records.
If Not (MySet.BOF = True) Then
'Go to the first record in the set of records returned by the query.
MySet.MoveFirst
'Loop through all records and write the fields into the kill table.
   While Not (MySet.EOF)
   MyRecords.AddNew
   MyRecords.Fields(0) = MySet.MSSN
   MyRecords.Fields(1) = MySet.pid
   MyRecords.Fields(2) = MySet.KTIME
   MyRecords.Fields(3) = MySet.Bumperno
   MyRecords.UPDATE
   MySet.MoveNext
   Wend
```

```
End If
End Sub
```

We also require a table of Blue vehicles. See Table 10 and the code below.

| MISSNUM | PID | PLAYER | | PLATFORM | FORCE |
|---|---|---|---|---|---|
| 008 | 19 | A65 | 1/A/3-67 AR | M1A1 | BL |
| 008 | 51 | A32 | 3/A/3-67 AR | M1A1 | BL |
| 008 | 53 | A11 | 2LT Burke/SGT Trowbridge | M1A1 | BL |
| 008 | 119 | A33 | 3/A/3-67 AR | M1A1 | BL |
| 008 | 205 | A31 | 3/A/3-67 AR | M1A1 | BL |
| 008 | 219 | A66 | 1/A/3-67 AR | M1A1 | BL |
| 008 | 235 | A34 | 3/A/3-67 AR | M1A1 | BL |
| 008 | 269 | A12 | SSG Burns/PFC Parker | M1A1 | BL |
| 008 | 1037 | A36 | 3/A/2-8 IN | M2 | BL |
| 008 | 1117 | A65 | 3/A/2-8 IN | M2 | BL |
| 008 | 1233 | A31 | 3/A/2-8 IN | M2 | BL |
| 008 | 1261 | A32 | 3/A/2-8 IN | M2 | BL |

Table 10. Blue identification table

```
'This routine produces a table of friendly vehicles as shown above.
Sub BuildBlue()
   Dim dbs As DATABASE
   Dim strSQL As String, qdf As QueryDef
   Dim MyRecords, MySet As Recordset
   Dim SQLquery As String
   Dim zeroes As String
   Dim miznum


'Return Database object pointing to current database.
 Set dbs = CurrentDb
'Check if Blue_ID_tbl exist.
'If table exist, then empty it.
If isTable("Blue_ID_tbl") Then
   strSQL = "Delete*from Blue_ID_tbl;"
   Set qdf = dbs.CreateQueryDef("", strSQL)
   qdf.Execute
End If
'Retrieve the mission number form the [SPECIFY MISSION FORM] which is up
and running once the user selects "initialize data tables."
miznum = [Forms]![SPECIFY MISSION_frm].[msnnum]
'This is a string fixing routine to put the mission number in the correct string
format.
If Len(miznum) = 2 Then
zeroes = "0"
```

```
Else
zeroes = "00"
End If
'Open Blue_ID Table.
Set MyRecords = dbs.OpenRecordset("Blue_ID_tbl")
'Populate the Blue_ID table with the fields shown below for tanks and bradleys
from the DBSCENAR table.

'Define the select query
SQLquery = "SELECT DISTINCTROW DBSCENAR.MISSNUM, DBSCENAR.PID,
DBSCENAR.PLAYER, DBSCENAR.PLATFORM, DBSCENAR.FORCE FROM
DBSCENAR WHERE (((DBSCENAR.MISSNUM)='" & zeroes & miznum & "') AND
((DBSCENAR.PLATFORM) Like '" & "M1A1" & "' Or (DBSCENAR.PLATFORM)
Like'" & "M2" & "'))"
'Run the query.
Set MySet = dbs.OpenRecordset(SQLquery)
'Check to see if the query produced zero records.
If Not (MySet.BOF = True) Then
'Go to the first record in the set of records returned by the query.
MySet.MoveFirst
'Loop through all records and write the fields into the Blue_ID table.
  While Not (MySet.EOF)
  MyRecords.AddNew
  MyRecords.Fields(0) = MySet.MISSNUM
  MyRecords.Fields(1) = MySet.pid
  MyRecords.Fields(2) = MySet.PLAYER
  MyRecords.Fields(3) = MySet.PLATFORM
  MyRecords.Fields(4) = MySet.FORCE
  MyRecords.UPDATE
  MySet.MoveNext
  Wend
End If
End Sub
```

Likewise we require a Red identification table. See Table 11 and the code below.

| MISSNUM | PID | PLAYER | PLATFORM | FORCE |
|---|---|---|---|---|
| 008 | 91 | A65 1/A/1-67 AR | T80 | OP |
| 008 | 251 | A11 1/A/1-67 AR | T80 | OP |
| 008 | 1269 | A21 2/A/1-67 AR | T80 | OP |
| 008 | 9 | A31 3/A/1-67 AR | T80 | OP |
| 008 | 1183 | A32 3/A/1-67 | T80 | OP |

Table 11. Red vehicle identification table.

```
'This routine produces a table of Red vehicles as shown above.
Sub BuildRed()
```

```
   Dim dbs As DATABASE
   Dim strSQL As String, qdf As QueryDef
   Dim MyRecords, MySet As Recordset
   Dim SQLquery As String
   Dim zeroes As String
   Dim miznum


'Return Database object pointing to current database.
 Set dbs = CurrentDb
'Check if Red_ID_tbl exist.
'If table exist, then empty it.
If isTable("Red_ID_tbl") Then
   strSQL = "Delete*from Red_ID_tbl;"
   Set qdf = dbs.CreateQueryDef("", strSQL)
   qdf.Execute
End If
'Retrieve the mission number form the [SPECIFY MISSION FORM] which is up
and running once the user selects "initialize data tables."
miznum = [Forms]![SPECIFY MISSION_frm].[msnnum]
'This is a string fixing routine to put the mission number in the correct string
format.
If Len(miznum) = 2 Then
zeroes = "0"
Else
zeroes = "00"
End If
'Open Red_ID_tbl Table.
Set MyRecords = dbs.OpenRecordset("Red_ID_tbl")

'Define the select query.
SQLquery = "SELECT DISTINCTROW DBSCENAR.MISSNUM, DBSCENAR.PID,
DBSCENAR.PLAYER, DBSCENAR.PLATFORM, DBSCENAR.FORCE FROM
DBSCENAR WHERE (((DBSCENAR.MISSNUM)='" & zeroes & miznum & "') AND
((DBSCENAR.PLATFORM) Like '" & "T80" & "') AND ((DBSCENAR.FORCE) Like'"
& "OP" & "'))"
'Run the query
Set MySet = dbs.OpenRecordset(SQLquery)
'Check to see if the query produced zero records.
If Not (MySet.BOF = True) Then
'Go to the first record in the set of records returned by the query.
MySet.MoveFirst
'Loop through all records and write the fields into the Red_ID table.
   While Not (MySet.EOF)
   MyRecords.AddNew
```

```
  MyRecords.Fields(0) = MySet.MISSNUM
  MyRecords.Fields(1) = MySet.pid
  MyRecords.Fields(2) = MySet.PLAYER
  MyRecords.Fields(3) = MySet.PLATFORM
  MyRecords.Fields(4) = MySet.FORCE
  MyRecords.UPDATE
  MySet.MoveNext
  Wend
End If
End Sub
```

The location table is a subset of the DBLOCATN table provided by the client. We query the DBLOCATN table for records pertaining to our selected mission. See Table 12 and the following code.

| MSN | PID | S_Time | E_Time | POSITION |
|---|---|---|---|---|
| 008 | 1037 | 7:25:25 AM | 7:25:35 AM | 15046797 |
| 008 | 1037 | 7:25:35 AM | 7:25:49 AM | 15066793 |
| 008 | 1037 | 7:25:49 AM | 7:26:05 AM | 15056792 |
| 008 | 1037 | 7:26:05 AM | 7:26:16 AM | 15166784 |
| 008 | 1037 | 7:26:16 AM | 7:27:04 AM | 15206781 |
| 008 | 1037 | 7:27:04 AM | 7:27:20 AM | 15226784 |
| 008 | 1037 | 7:27:20 AM | 7:27:46 AM | 15246784 |
| 008 | 1037 | 7:27:46 AM | 7:29:02 AM | 15246785 |
| 008 | 1037 | 7:29:02 AM | 7:29:13 AM | 15256785 |
| 008 | 1037 | 7:29:13 AM | 7:29:26 AM | 15266782 |
| 008 | 1037 | 7:29:26 AM | 7:29:39 AM | 15286777 |
| 008 | 1037 | 7:29:39 AM | 7:29:58 AM | 15336769 |
| 008 | 1037 | 7:29:58 AM | 7:30:11 AM | 15396760 |

Table 12. Sample of the location table. The table shows where vehicles were located during a time window between start time (S_Time) and end time (E_Time).

```
'This routine builds the location table by taking the subset of records from
DBLOCATN pertaining to the user selected mission.
Sub BuildLocation()

Dim j, counter As Integer
Dim MyDb As DATABASE, OldRecords, NewRecords As Recordset
Dim strSQL As String, qdf As QueryDef
Dim SQLquery As String
Dim zeroes As String
Dim miznum


'Return Database object pointing to current database.
Set MyDb = DBEngine.Workspaces(0).Databases(0)
```

```
Set NewRecords = MyDb.OpenRecordset("Location")

'Check if Location table exist.
'If table exist, then empty it.
If isTable("Location") Then
   strSQL = "Delete*from Location;"
   Set qdf = MyDb.CreateQueryDef("", strSQL)
   qdf.Execute
End If
'Retrieve the mission number form the [SPECIFY MISSION FORM] which is up
and running once the user selects "initialize data tables."
miznum = [Forms]![SPECIFY MISSION_frm].[msnnum]
'This is a string fixing routine to put the mission number in the correct string
format.
If Len(miznum) = 2 Then
zeroes = "0"
Else
zeroes = "00"
End If

'Open Location Table.
Set NewRecords = MyDb.OpenRecordset("Location")
'Retrieve the records that pertain to the mission selected in the SPECIFY MISSION
form from the DBLOCATN table.

'Define the select query.
SQLquery = "SELECT DISTINCTROW DBLOCATN.SITE, DBLOCATN.CELL,
DBLOCATN.MISSNUM, DBLOCATN.DATE, DBLOCATN.PID,
DBLOCATN.PLAYER, DBLOCATN.S_TIME, DBLOCATN.E_TIME,
DBLOCATN.POSITION FROM DBLOCATN WHERE (((DBLOCATN.MISSNUM)='"
& zeroes & miznum & "'))"
'Run the query.
Set OldRecords = MyDb.OpenRecordset(SQLquery)
'Now convert the selected data in DBSCENARIO to usable form in Location table.
'Go to the first record in the set of records returned by the query.
OldRecords.MoveFirst
'Loop through all records and write the converted data into the fields of the location
table.
While OldRecords.EOF = False
NewRecords.AddNew
NewRecords.Fields(0) = OldRecords.MISSNUM
NewRecords.Fields(1) = OldRecords.pid
NewRecords.Fields(2) = parse_time(OldRecords.S_Time) 'converts into the right time
format
NewRecords.Fields(3) = parse_time(OldRecords.E_Time)
```

```
NewRecords.Fields(4) = CLng(gridparse(OldRecords.POSITION)) 'converts grid
coordinates to long integer format.
NewRecords.UPDATE
OldRecords.MoveNext
Wend

End Sub
```

**The FixData routine prunes the data of outliers. The first part queries the data for vehicles that have no entries in the location table. These vehicles are automatically removed from the Blue_ID table. By observing output reports which we developed to show individual vehicle movement we were able to spot vehicles that did not move or moved out of sector during the battle. We *hardwired* the pruning of these vehicles so that the data would be clean on subsequent runs of the particular missions.**

```
Sub FixData()
Dim dbs As DATABASE
Dim strSQL As String, qdf As QueryDef
Dim MyRecords, MySet As Recordset
Dim SQLquery As String

Set dbs = CurrentDb
'Some of the blue vehicles do not show up in the location table (most likely due to
maintenance or lack of instrumentation.
'Edit Blue_ID_tbl of entries that have no position data. We do this because we can
not know ground truth about these vehicles' movements during the battle.
SQLquery = "SELECT DISTINCT Blue_ID_tbl.PID FROM Blue_ID_tbl LEFT JOIN
Location ON Blue_ID_tbl.PID = Location.PID WHERE (((Location.POSITION) Is
Null))"
'Run the query.
Set MyRecords = dbs.OpenRecordset(SQLquery)
'Open blue ID table.
Set MySet = dbs.OpenRecordset("Blue_ID_tbl")
'Check to see if the query produced zero records.
If Not (MyRecords.BOF) Then
'Move to first record in the query record set.
MyRecords.MoveFirst
'Loop through all the Blue ID's that are not listed in the location data.
Do Until MyRecords.EOF
'Go to first entry of the Blue ID table.
   MySet.MoveFirst
'Loop through all entries of the Blue ID table.
   Do Until MySet.EOF
'If the vehicle ID in the Blue ID table matches one of the vehicle ID's in the query
then delete it from the Blue ID table.
```

```
    If MySet![pid] = MyRecords![pid] Then
       MySet.Delete
    End If
    MySet.MoveNext
    Loop
MyRecords.MoveNext
Loop
MySet.Close
MyRecords.Close
End If
```

**'Other vehicles may have entries in the location data but move way out of sector or do not move at all. We delete these vehicles because they do not participate in the battle and skew the computation of center of mass of the unit.**
**'Cleanse particular vehicle entries for particular missions.**

```
Set MySet = dbs.OpenRecordset("Blue_ID_tbl")
If Forms![SPECIFY MISSION_frm].[msnnum] = 16 Then
  MySet.MoveFirst
   Do Until MySet.EOF
   If MySet![pid] = "35" Then
      MySet.Delete
   End If
   MySet.MoveNext
   Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 12 Then
  MySet.MoveFirst
   Do Until MySet.EOF
   If MySet![pid] = "119" Then
      MySet.Delete
   End If
   MySet.MoveNext
   Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 14 Then
  MySet.MoveFirst
   Do Until MySet.EOF
   If (MySet![pid] = "241" Or MySet![pid] = "105") Then
      MySet.Delete
   End If
   MySet.MoveNext
   Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 10 Then
  MySet.MoveFirst
   Do Until MySet.EOF
   If MySet![pid] = "79" Then
      MySet.Delete
   End If
```

```
    MySet.MoveNext
    Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 3 Then
   MySet.MoveFirst
    Do Until MySet.EOF
    If MySet![pid] = "83" Then
       MySet.Delete
    End If
    MySet.MoveNext
    Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 6 Then
   MySet.MoveFirst
    Do Until MySet.EOF
    If MySet![pid] = "41" Then
       MySet.Delete
    End If
    MySet.MoveNext
    Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 7 Then
   MySet.MoveFirst
    Do Until MySet.EOF
    If MySet![pid] = "99" Then
       MySet.Delete
    End If
    MySet.MoveNext
    Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 9 Then
   MySet.MoveFirst
    Do Until MySet.EOF
    If MySet![pid] = "99" Or MySet![pid] = "105" Or MySet![pid] = "157" Or MySet![pid]
= "173" Or MySet![pid] = "1037" Or MySet![pid] = "1255" Then
       MySet.Delete
    End If
    MySet.MoveNext
    Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 20 Then
   MySet.MoveFirst
    Do Until MySet.EOF
    If MySet![pid] = "13" Or MySet![pid] = "105" Or MySet![pid] = "35" Or MySet![pid]
= "83" Or MySet![pid] = "1261" Then
       MySet.Delete
    End If
    MySet.MoveNext
    Loop
ElseIf Forms![SPECIFY MISSION_frm].[msnnum] = 8 Then
```

```
  MySet.MoveFirst
   Do Until MySet.EOF
   If MySet![pid] = "37" Or MySet![pid] = "105" Or MySet![pid] = "157" Or MySet![pid]
= "83" Or MySet![pid] = "241" Or MySet![pid] = "1021" Or MySet![pid] = "1143" Then
      MySet.Delete
   End If
   MySet.MoveNext
   Loop
MySet.Close
End If

End Sub
```

**'This routine sets up the data tables needed for our computations as discussed above. When the user selects "Initialize data tables" from the menu this routine opens the select mission dialog box which awaits a selection from the user. Once the user makes his/her selection the routine builds the tables listed, cleanses the data of outliers and builds the true detections table (discussed below). The routine then opens the Flot_frm dialog which allows the user to view the individual movements of all Blue vehicles which are represented by vectors.**

```
Sub Initialize()
DoCmd.OpenForm "SPECIFY MISSION_frm", , , , , acDialog
DoCmd.Hourglass True
BuildLocation
BuildKill
BuildBlue
BuildRed
FixData
TrueTable
DoCmd.OpenForm "Flot_frm", , , , , acDialog

End Sub
```

DETECTIONS / FRATRICIDE INDEX

| Red PID | Red Time | Red | Missed | Blue PID | Blue Time | Blue Position |
|---|---|---|---|---|---|---|
| 1183 | 7:49:00 AM | 16686624 | 178.8854 | 119 | 7:47:51 AM | 16536694 |
| 251 | 7:12:03 AM | 16516618 | 2103.449 | 19 | 7:12:03 AM | 14446771 |
| 251 | 7:12:03 AM | 16516618 | 1251.599 | 19 | 7:13:51 AM | 14776771 |
| 251 | 7:26:05 AM | 16506599 | 10 | 19 | 7:28:02 AM | 15736647 |
| 9 | 7:55:41 AM | 17416692 | 245.9675 | 235 | 7:58:08 AM | 17146710 |
| 91 | 8:22:14 AM | 16996640 | 319.5309 | 53 | 8:22:34 AM | 16896695 |
| 9 | 8:25:44 AM | 17406689 | 214.7091 | 53 | 8:27:11 AM | 17246713 |

Tables 13a and 13b. Detection data showing the nearest Red vehicle and nearest Blue vehicle to the detection location, the distance between these two vehicles and the resulting fratricide index.

| Blue Radius | Dist R to B | Frat Index | Reported |
|---|---|---|---|
| 544.5181 | 715.8911 | 0.5107379 | 4 |
| 911.9759 | 2574.063 | -0.4628766 | 1 |
| 1320.984 | 2317.002 | 2.994623E-02 | 2 |
| 902.1086 | 907.3588 | 0.9831928 | 3 |
| 430.8132 | 324.4996 | 0.569633 | 5 |
| 326.4966 | 559.017 | 1.246051E-02 | 6 |
| 434.1659 | 288.4441 | 0.7608294 | 7 |

Table 13b.

```
'Declarations
Option Compare Database
Option Explicit

Global NumReported As Integer

'This routine checks each reported detection in the Detect table to determine which
enemy vehicle was actually detected and the accuracy of the detection. Results are
recorded in the Detect_True table.

Sub TrueTable()
Dim MyDb As DATABASE, MySet As Recordset, SQLRedquery, SQLBluequery As
String
Dim RedQuery, BlueQuery As Recordset
Dim Tdelta As String
Dim i, counter As Integer
Dim MyTime, newtime
Dim strSQL As String, qdf As QueryDef
Dim MyLocation
Dim SQLquery As String
Dim MyRecords As Recordset
```

**'Set the time increment to five minutes. This provides a time window in which the detection is checked which helps account for transmission latency.**

Tdelta = 5

**'Return Database object pointing to current database.**

Set MyDb = DBEngine.Workspaces(0).Databases(0)

**'Open the Detect_True table for writing.**

Set MySet = MyDb.OpenRecordset("Detect_True_tbl")

**'Check if Detect_True_tbl exist.**

**'If the table already exist, then empty it.**

If isTable("Detect_True_tbl") Then

strSQL = "Delete*from Detect_True_tbl;"

Set qdf = MyDb.CreateQueryDef("", strSQL)

qdf.Execute

End If

**'Find the records in the Detect_tbl that pertain to the mission selected in the SPECIFY MISSION form.**

SQLquery = "SELECT DISTINCTROW Detect_tbl.MSN, Detect_tbl.Time, Detect_tbl.Enemy, Detect_tbl.Location, Detect_tbl.Source FROM Detect_tbl WHERE (((Detect_tbl.MSN)='" & [Forms]![SPECIFY MISSION_frm].[msnnum] & "'))"

**'Run the query.**

Set MyRecords = MyDb.OpenRecordset(SQLquery)

MyRecords.MoveLast

**'Find the number of reported detections for this mission i.e. number of records in the record set returned by the query.**

NumReported = MyRecords.RecordCount

i = 0

counter = 0

**'Move to the first reported detection.**

MyRecords.MoveFirst

**'Loop through detections until all have been checked.**

While MyRecords.EOF = False

counter = counter + 1

**'Set a variable to hold the time of detection.**

MyTime = MyRecords.[Time]

**'Set a variable to hold the location of detection.**

MyLocation = MyRecords.[Location]

**'Add a new record to the Detect_True table.**

MySet.AddNew

**'Subtract 5 minutes from the time of detection. Call it "newtime."**

newtime = DateAdd("n", -Tdelta, MyTime)

**'Define a query to select the nearest red vehicle vicinity the detection location within a time window of five minutes prior to detection up to detection time.**

```
   SQLRedquery = "SELECT DISTINCTROW TOP 1 Red_ID_tbl.PID,
Location.E_Time, Location.Position, distance('" & MyLocation & "',[Position]) AS
Distance FROM Red_ID_tbl INNER JOIN Location ON Red_ID_tbl.PID = Location.PID
WHERE (((Location.E_Time) Between #" & newtime & "# And #" & MyTime & "#))
ORDER BY distance('" & MyLocation & "',[Position]);"
'Likewise find the nearest Blue vehicle for fratricide computations.
   SQLBluequery = "SELECT DISTINCTROW TOP 1 Blue_ID_tbl.PID,
Location.E_Time, Location.Position, distance('" & MyLocation & "',[Position]) AS
Distance FROM Blue_ID_tbl INNER JOIN Location ON Blue_ID_tbl.PID =
Location.PID WHERE (((Location.E_Time) Between #" & newtime & "# And #" &
MyTime & "#)) ORDER BY distance('" & MyLocation & "',[Position]);"
'Run the two queries.
   Set RedQuery = MyDb.OpenRecordset(SQLRedquery)
   Set BlueQuery = MyDb.OpenRecordset(SQLBluequery)
'Check for detections that produced no enemy and report the data to the analyst in
a message box.
   If (RedQuery.BOF = True) Then
   MsgBox (newtime & "    " & MyTime & "    " & MyLocation)
   End If
'Otherwise populate the Detect_True table with the following fields:
   If Not (RedQuery.BOF = True) Then
      MySet.Fields(0) = RedQuery.Fields(0) 'Red vehicle ID (Pid)
      MySet.Fields(1) = RedQuery.Fields(1) 'Red Time of detection
      MySet.Fields(2) = RedQuery.Fields(2) 'Red position
      MySet.Fields(3) = RedQuery.Fields(3) 'Red missed radius (How far off was the
reported grid?)
      MySet.Fields(4) = BlueQuery.Fields(0) 'Blue Pid
      MySet.Fields(5) = BlueQuery.Fields(1) 'Blue Time of detection
      MySet.Fields(6) = BlueQuery.Fields(2) 'Blue position
      MySet.Fields(7) = BlueQuery.Fields(3) 'Blue missed radius
'Find the distance from red vehicle to blue vehicle.
      MySet.Fields(8) = Distance(RedQuery.Fields(2), BlueQuery.Fields(2))
'Compute the fratricide index. (f-r)/d where f is the distance from the nearest blue
vehicle and r is the distance from the nearest red vehicle and d is the distance
between the two vehicles.
      MySet.Fields(9) = (BlueQuery.Fields(3) - RedQuery.Fields(3)) / MySet.Fields(8)
MySet.Fields(10) = counter
      MySet.UPDATE

   End If
'Go to next reported detection.
MyRecords.MoveNext
Wend
DoCmd.Hourglass False
End Sub
```

| Category | Count |
|---|---|
| -1 -0.75 | 0 |
| -0.75 -0.5 | 0 |
| -0.5 -0.25 | 1 |
| -0.25 0 | 0 |
| 0 0.25 | 2 |
| 0.25 0.5 | 0 |
| 0.5 0.75 | 2 |
| 0.75 1 | 2 |

Table 14. Fratricide index values for the detections during a particular mission organized by bins.

```
'This routine organizes the fratricide data computed in the TrueTable routine for
display as a histogram chart.
Sub Fratricide()
Dim MyDb As DATABASE, MySet As Recordset, SQLquery As String
Dim i, counter As Integer
Dim strSQL As String, qdf As QueryDef
Dim MyRecords As Recordset
Dim LowerBound, UpperBound As Single

Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MySet = MyDb.OpenRecordset("Frat_tbl")


'Clear the Fratricide table.
If isTable("Frat_tbl") Then
strSQL = "Delete*from Frat_tbl;"
Set qdf = MyDb.CreateQueryDef("", strSQL)
qdf.Execute
End If
'Record column names in the frat table.
MySet.AddNew
MySet.Fields(0) = "BINS
MySet.Fields(1) = 0
MySet.UPDATE

'Initialize the bin boundaries i.e. first bin is [-1,-.75].
LowerBound = -1
UpperBound = -0.75
For i = 1 To 8
'Select the fratride entries that fall within this bin range.
SQLquery = "SELECT DISTINCTROW Detect_True_tbl.[Frat Index] FROM
Detect_True_tbl WHERE (((Detect_True_tbl.[Frat Index]) <" & UpperBound & "And
(Detect_True_tbl.[Frat Index]) >" & LowerBound & "))"
```

```
'Run the query.
Set MyRecords = MyDb.OpenRecordset(SQLquery)
'If there are no fratricide index values in this range then zero is recorded in the
table.
If MyRecords.BOF Then
   MySet.AddNew
   MySet.Fields(0) = LowerBound & "  " & UpperBound
   MySet.Fields(1) = 0
   MySet.UPDATE
'Set boundaries for the next bin.
   LowerBound = LowerBound + 0.25
   UpperBound = UpperBound + 0.25
Else
'Go to the last record so the RecordCount function can be used.
MyRecords.MoveLast
'Prepare the fratricide table for another entry.
MySet.AddNew
MySet.Fields(0) = LowerBound & "  " & UpperBound
'Count the number of index values in the range of this bin.
MySet.Fields(1) = MyRecords.RecordCount
MySet.UPDATE
'Set boundaries for the next bin.
LowerBound = LowerBound + 0.25
UpperBound = UpperBound + 0.25
End If
Next
'Close the Flot_frm which has remained open but invisible.
DoCmd.Close acForm, "Flot_frm", acSaveYes
End Sub
```

## COMPUTE MOVEMENT VECTORS

With data tables established the software next computes the individual and unit movement vectors. This is triggered by the Flot_frm dialog box which appears once the data tables are initialized. We use the individual vehicle movement vectors to visually check for outlier vehicles which either fail to move or move off out of the battle area (possibly due to maintenance problems). As discussed earlier in the report, we use the unit movement vectors to compute the area searched or swept by the attacking unit.

```
'Declarations: Note the many global variables that this routine generates.
Option Compare Database
Option Explicit
Global Vector(), Variance(), Normal(), Areas(), CMx(), CMy(), Magnitude() As Double
Global Movement() As Variant
```

```
Global k, j As Integer
Global MaxX, MaxY, MinX, MinY As Single
Global deltaT As String

'This routine builds the movement array which is a global variable holding the
movement vectors of each Blue vehicle over time as well as all other global variables
and arrays listed above in the declarations.

Sub MovementArray()
DoCmd.Hourglass True
Dim Distance(), NormDist(), CumX(), CumY(), Maxlength(), Minlength() As Single
Dim MyDb As DATABASE, MyRecords, BlueQuery, PositionQuery As Recordset,
SQLquery, SQL2query As String
Dim CurrentVeh As String
Dim SumSqrDist, SumNormDist As Single
Dim Time, Length
Dim i, t, m, num, r, v As Integer
'Gets the time increment from the user who selects it from the Flot_frm dialog box.
deltaT = Forms![Flot_frm].[deltat_txt]
Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MyRecords = MyDb.OpenRecordset("Blue_ID_tbl")

MyRecords.MoveLast
'Count the number of Blue vehicles in the Blue ID table.
j = MyRecords.RecordCount
'Move to first vehicle in the Blue_ID table.
MyRecords.MoveFirst
'Set a counter.
num = 1
'Find how many minutes of battle time between the start time of the battle and the
end time of the battle. Divide this value by the time step selected by the user to get
the number of such time steps (k). Start times and end times are automatically
displayed for the user in the Flot_frm dialog box.
k = Fix((DateDiff("n", Forms![Flot_frm].[Start Time_txt], Forms![Flot_frm].[End
Time_txt])) / deltaT)
'Dimension the arrays to appropriate sizes based on the number of vehicles (j) and
the number of time steps (k).
ReDim Movement(1 To j, k + 1), Vector(1 To 2, 1 To k), Normal(1 To 2, 1 To k + 1),
Magnitude(1 To k + 1), Maxlength(k), Minlength(k)
ReDim Distance(1 To 2, 1 To j, k), NormDist(1 To j, k), Variance(k)
ReDim Areas(k), CumX(k), CumY(k), CMx(k), CMy(k)
'Set max and min boundaries for later comparisons to determine Max and Min
distances of vehicles from the unit center of mass.
MinX = 54000
MaxX = 0
```

MaxY = 0
MinY = 54000
**'Loop through the Blue vehicles.**
While MyRecords.EOF = False
**'Variable holding the current vehicle's ID.**
CurrentVeh = MyRecords.[pid]
**'Variable holding the current time in the battle.**
Time = Forms![Flot_frm].[Start Time_txt]
**'Fill the last column of the Movement array with the vehicle ID.**
Movement(num, k + 1) = CurrentVeh

For i = 0 To k
**'Find the location of the current vehicle for each time step. Where "time" falls between Start time and End time.**
**'The variable "time" will be updated with each pass through the loop.**
SQLquery = "SELECT DISTINCT Location.Position FROM Location WHERE (((Location.PID)='" & CurrentVeh & "')AND ((Location.S_Time)<=#" & Time & "#)AND ((Location.E_Time)>=#" & Time & "#));"
Set BlueQuery = MyDb.OpenRecordset(SQLquery)
**'If the query returns no record than either the position location instrumentation was turned off early or turned on after start time.**
**'We run another query to find the first reported location of the vehicle in the battle and begin with this location (SQL2query).**
If (BlueQuery.BOF) Then
If (i = 0) Then
SQL2query = "SELECT DISTINCTROW TOP 01 Location.PID, Location.POSITION FROM Location WHERE (((Location.PID)='" & CurrentVeh & "'))"
Set PositionQuery = MyDb.OpenRecordset(SQL2query)
**'If this corresponds to the first time step we set the position equal to this first reported (known) location. Otherwise we simply use the last known position of the vehicle ("Movement(num, i-1)").**
Movement(num, i) = PositionQuery.Fields(1)
Else
Movement(num, i) = Movement(num, i - 1)
End If
Else
**'If the query does return a location we begin with this location.**
Movement(num, i) = BlueQuery.Fields(0)
End If
**'Since we know we are going to want to know which vehicle is farthest form the unit center of mass in the +normal and -normal directions from the CM movement vector, we keep a running tally of the Max and Min x and y values.**
MaxX = Max(MaxX, CInt(Mid(Movement(num, i), 1, (Len(Movement(num, i)) - 4)))) 'in 10m units
MinX = Min(MinX, CInt(Mid(Movement(num, i), 1, (Len(Movement(num, i)) - 4))))

```
    MaxY = Max(MaxY, CInt(Mid(Movement(num, i), (Len(Movement(num, i)) - 3))))
    MinY = Min(MinY, CInt(Mid(Movement(num, i), (Len(Movement(num, i)) - 3))))

'Likewise we accumulate xcoord in CumX and ycoord in CumY so that we can
compute center of mass.
    CumX(i) = CumX(i) + CInt(Mid(Movement(num, i), 1, (Len(Movement(num, i)) -
4)))
    CumY(i) = CumY(i) + CInt(Mid(Movement(num, i), (Len(Movement(num, i)) - 3)))
'Update the time by adding another time increment to the current time.
    Time = DateAdd("n", deltaT, Time)
   Next i
'Update the counter of number of which vehicle we are working on.
   num = num + 1
'Get the next vehicle.
   MyRecords.MoveNext
Wend
'Since we have now gone through all vehicles for this particular time step we can
compute center of mass.
'Store Center Of Mass in CMx and CMy arrays and initialize max length and min
length.
For i = 0 To k
    CMx(i) = CumX(i) / j  'xcm
    CMy(i) = CumY(i) / j 'ycm
    Maxlength(i) = 0
    Minlength(i) = 54000
Next i

'Compute and store vector values and normal vector values and fill the magnitude
array.
For v = 1 To k
  Vector(1, v) = CMx(v) - CMx(v - 1) 'x coordinate difference
  Vector(2, v) = CMy(v) - CMy(v - 1) 'y coordinate difference
  Normal(1, v) = -Vector(2, v)  'Normalx = -Vectory
  Normal(2, v) = Vector(1, v)   'Normaly = Vectorx
'Compute the magnitude of the unit movement vector.  We start at v = 1 since we
need two points from which to determine x and y differences.
  Magnitude(v) = Sqr((Normal(1, v) ^ 2) + (Normal(2, v) ^ 2)) 'in 10m units
  Next v

'Determine distances from CM in the normal direction.
num = j
'Set final normal vector so length calculation do not stumble on k+1.
Normal(1, k + 1) = Normal(1, k)
Normal(2, k + 1) = Normal(2, k)
Magnitude(k + 1) = Magnitude(k)
```

**'For each time step.**
For t = 0 To k
**'For each vehicle.**
  For m = 1 To num
**'X-coord of difference from vehicle to CM.**
  Distance(1, m, t) = (CInt(Mid(Movement(m, t), 1, (Len(Movement(m, t)) - 4))) - CMx(t)) 'deltax in 10m units
**'Y-coord of difference from vehicle to CM.**
  Distance(2, m, t) = (CInt(Mid(Movement(m, t), (Len(Movement(m, t)) - 3))) - CMy(t)) 'deltay in 10m units

**'Now project distance onto the Normal by dot product/mag(Normal)check for maxlength and minlength and store in lengths(1,)and lengths(2,)respectively.**
**'Check for division by zero.**
  If Magnitude(t + 1) = 0 Then
    If t = 0 Then
    NormDist(m, t) = 0
    Else
    NormDist(m, t) = NormDist(m, t - 1)
    End If
  Else
**'Length is distance in the Normal direction. Compute Length and compare with Max and Min distance from CM.**
  Length = ((Distance(1, m, t) * Normal(1, t + 1) + Distance(2, m, t) * Normal(2, t + 1)) / Magnitude(t + 1))
**'Store Normal Distances for use in the reports section (graphics).**
  NormDist(m, t) = Length
  Maxlength(t) = Max(Length, Maxlength(t)) 'furtherest vehicle above CM
  Minlength(t) = Min(Length, Minlength(t)) 'furtherest vehicle below CM
  End If

  Next m
Next t
**'Compute Area swept during each time step.**
For t = 0 To k - 1
**'Add 750meters to each boundary in order to account for peripheral vehicles scanning beyond their locations.**
Areas(t + 1) = (Abs(Maxlength(t) * 10) + Abs(Minlength(t) * 10) + 1500) * (Magnitude(t + 1) * 10)
Next t
**'Compute variance of these normal distances from the CM. This information will be use to report on the unit's lateral deployment over time measured in standard deviations from the CM.**
For t = 0 To k
SumSqrDist = 0

```
SumNormDist = 0
   For m = 1 To j
   'Sum of Squared Distances
   SumSqrDist = SumSqrDist + NormDist(m, t) ^ 2
   'Sum of Distances Squared
   SumNormDist = SumNormDist + NormDist(m, t)
   Next m
   Variance(t) = (SumSqrDist - (SumNormDist ^ 2 / j)) / (j - 1)
Next t
```

**'User Interface: Turn off Hourglass graphic.**

```
DoCmd.Hourglass False
```

**'Open the Step form which allows the user to visually observe the units' movement as represented by individual vehicle movement vectors for a user-determined number of time steps.**

```
DoCmd.OpenForm "Step_frm", acNormal, "", "", acEdit, acNormal
```

**'User interface: Turns off the Flot form so the user can not see it. Form is still up and running.**

```
Forms!Flot_frm.Visible = False

End Sub
```

ENTROPY AND INFO GAIN

**'Declarations**

```
Option Compare Database
Option Explicit

Global Uncertain(), Gain() As Single
Global MaxE As Single
```

**'This routine computes the entropy concerning each enemy vehicle at every time step, computes information gain from this data as the change in entropy for each vehicle and stores the individual values as well as cumulative values in the Info_Gain table. The routine also computes and stores normalized values in the NormEntropy table.**

```
Sub ComputeE()
DoCmd.Hourglass True
Dim MyDb As DATABASE, MyRecords, Detquery, Info, Norm As Recordset,
SQLquery1, SQLquery2 As String
Dim i, l, m, num, r, v, jnum, red As Integer
Dim dtime()
Dim state() As String
Dim dradius() As Single
```

```
Dim Areatot As Single
Dim starttime, oldtime, newtime, TimeStep, Rate
Dim strSQL As String, qdf As QueryDef
'Dim MaxE As Single
Dim localMaxE As Single

Set MyDb = DBEngine.Workspaces(0).Databases(0)
'Open the Red_ID table.
Set MyRecords = MyDb.OpenRecordset("Red_ID_tbl")
'Open the Info_Gain table.
Set Info = MyDb.OpenRecordset("Info_Gain_tbl")
'Open the NormEntropy table.
Set Norm = MyDb.OpenRecordset("NormEntropy_tbl")

'Clear the info table.
If isTable("Info_Gain_tbl") Then
strSQL = "Delete*from Info_Gain_tbl;"
Set qdf = MyDb.CreateQueryDef("", strSQL)
qdf.Execute
End If

'Clear the NormEntropy_tbl table.
If isTable("NormEntropy_tbl") Then
strSQL = "Delete*from NormEntropy_tbl;"
Set qdf = MyDb.CreateQueryDef("", strSQL)
qdf.Execute
End If

'Find number of Red Vehicles listed in the Red_ID table using the RecordCount
function.
MyRecords.MoveLast
red = MyRecords.RecordCount
'Retrieve the start time of the battle from the Flot Form.
starttime = Forms![Flot_frm].[Start Time_txt]
'Compute the total battlefield area from the global variables MaxX and MaxY
whose values were generated in the MovementArray routine. The 500 is added for
sensor ability to scan beyond the vehicles' physical location.
Areatot = (((MaxX - MinX) * 10 + 500) * ((MaxY - MinY) * 10 + 500))
'Set the movement rate of enemy vehicles. This is used in the degradation model.
Rate = 3 'km/hr

'Compute Maximum Entropy for use in Normalization of Entropy values.
localMaxE = -Log(Areatot)
'Maximum Entropy is the number of enemy vehicles times the entropy for one
vehicle.
```

```
MaxE = red * localMaxE
'Dimension the arrays of appropriate size.
ReDim Uncertain(1 To (red + 1), 0 To k), Gain(1 To (red + 1), 0 To k)
ReDim dtime(1 To red)
ReDim state(1 To red)
ReDim dradius(1 To red)
'For all enemy vehicles:
'Initialize initial entropy Uncertain[0]to max entropy. Uncertain[1] will be entropy
after 1st time step. Initialize initial detection time at time 0. Initialize initial state as
area calculation. Initialize initial detection radius to zero for each vehicle.
For l = 1 To red
  dtime(l) = 0
  state(l) = "Area"
  dradius(l) = 0
Next l
'Loop through each time step.
For i = 0 To k
'Decrement total area by the amount swept out by the attacking unit. This is from
the global array Areas which was produced in the MovementArray routine.
  Areatot = Areatot - Areas(i)
  'Guard against ln(0).
    If (Areatot <= 0) Then
    Areatot = 1#
    End If
  oldtime = DateAdd("n", (i - 1) * deltaT, starttime)
'Update the current time by multiplying deltaT by the number of time steps.
  newtime = DateAdd("n", (i) * deltaT, starttime)
'Go to first vehicle.
  MyRecords.MoveFirst
  jnum = 1
'Loop through all vehicles.
  While MyRecords.EOF = False
'This query checks to see if the current vehicle was detected during the current time
period. If detected more than once the query returns the most accurate detection i.e.
records are ordered by missed -distance and the query selects the first entry for the
current vehicle.
  SQLquery1 = "SELECT DISTINCTROW TOP 1 Detect_True_tbl.[Red PID],
Detect_True_tbl.[Red Time], Detect_True_tbl.[Missed Radius] FROM Detect_True_tbl
WHERE (((Detect_True_tbl.[Red PID]) Like '" & MyRecords.pid & "') AND
((Detect_True_tbl.[Red Time]) Between #" & oldtime & "# And #" & newtime & "#))
ORDER BY Detect_True_tbl.[Missed Radius];"
'Run the query.
  Set Detquery = MyDb.OpenRecordset(SQLquery1)
'Compute the entropy based on the state of the vehicle at each time step. Store these
values in the Uncertain array.
```

```
Select Case state(jnum)

    'If the enemy vehicle is in the "Area" state:
    Case Is = "Area"
    'If killed then entropy goes to zero.
    If (killtime(MyRecords.pid) < newtime) Then
    Uncertain(jnum, i) = 0
    'Change state of vehicle to "Kill."
    state(jnum) = "Kill"
    'If vehicle was detected.
    ElseIf Not (Detquery.BOF) Then
    'Entropy goes to log min((circular area with radius missed distance),
    log(areatot)).
    Uncertain(jnum, i) = -Log(Min(3.14159 * Detquery.Fields(2) ^ 2, Areatot))
    'Store detection time and detection radius for degradation calculations.
    dtime(jnum) = Detquery.Fields(1)
    dradius(jnum) = Detquery.Fields(2)
    'Change the state of the vehicle to "Detect."
    state(jnum) = "Detect"
    Else
    'Remain in Area state and compute entropy based on area searched.
    Uncertain(jnum, i) = -Log(Areatot)
    End If

    'If the enemy vehicle is in the "Detect" state:
    Case Is = "Detect"
    'Compute the time that has elapsed since detection in hours.
    TimeStep = DateDiff("s", dtime(jnum), newtime) / 60 / 60
    'If killed then entropy goes to zero.
    If (killtime(MyRecords.pid) < newtime) Then
    Uncertain(jnum, i) = 0
    'Change state of vehicle to "Kill."
    state(jnum) = "Kill"
    'If vehicle was detected again.
    ElseIf Not (Detquery.BOF) Then
    'Entropy goes to log min((circular area with radius missed distance),
    log(areatot)).
    Uncertain(jnum, i) = -Log(Min(3.14159 * Detquery.Fields(2) ^ 2, Areatot))
    'Store detection time and detection radius for degradation calculations.
    dtime(jnum) = Detquery.Fields(1)
    dradius(jnum) = Detquery.Fields(2)
    'Check to see if vehicle's entropy has degraded to the point that it is as if it
    were still in the "Area" state. If so then change state back to "Area."
    ElseIf ((3.14159 * (dradius(jnum) ^ 2 + (Rate * 1000 * TimeStep) ^ 2)) >=
    Areatot) Then
```

```
'Transition back to Area state.
Uncertain(jnum, i) = -Log(Areatot)
state(jnum) = "Area"
'Otherwise,'transition to "Degrade" state and compute entropy based on
increased radius over time.
Else
Uncertain(jnum, i) = -Log(3.14159 * (dradius(jnum) ^ 2 + (Rate * 1000 *
TimeStep) ^ 2))
state(jnum) = "Degrade"
End If

'If the enemy vehicle is in the "Degrade" state:
Case Is = "Degrade"
'Compute the time that has elapsed since detection in hours.
TimeStep = DateDiff("s", dtime(jnum), newtime) / 60 / 60
'If killed then entropy goes to zero.
If (killtime(MyRecords.pid) < newtime) Then
Uncertain(jnum, i) = 0
'Change state of vehicle to "Kill."
state(jnum) = "Kill"
'If vehicle was detected again.
ElseIf Not (Detquery.BOF) Then
'Entropy goes to log min((circular area with radius missed distance),
log(areatot)).
Uncertain(jnum, i) = -Log(Min(3.14159 * Detquery.Fields(2) ^ 2, Areatot))
'Store detection time and detection radius for degradation calculations.
dtime(jnum) = Detquery.Fields(1)
dradius(jnum) = Detquery.Fields(2)
'Change the state of the vehicle to "Detect."
state(jnum) = "Detect"
'If Detection area has degradeded to the size of remaining area.
ElseIf ((3.14159 * (dradius(jnum) ^ 2 + (Rate * 1000 * TimeStep) ^ 2)) >=
Areatot) Then
'Transition back to Area state.
Uncertain(jnum, i) = -Log(Areatot)
state(jnum) = "Area"
'Otherwise remain in the "Degrade" state.
Else
Uncertain(jnum, i) = -Log(3.14159 * (dradius(jnum) ^ 2 + (Rate * 1000 *
TimeStep) ^ 2))
End If

Case Else
'Vehicle must be dead.
  Uncertain(jnum, i) = 0
```

```
  End Select

 'Add results to the Info Table.
 Info.AddNew
 Info.Fields(0) = i
 Info.Fields(1) = newtime
 Info.Fields(2) = MyRecords.pid
 Info.Fields(3) = state(jnum)
 Info.Fields(4) = -Uncertain(jnum, i)
 If i = 0 Then
'There is no information gain at time zero since it is a difference measure of
entropy.
 Gain(jnum, i) = 0
 Info.Fields(5) = 0
 Info.Fields(9) = 0
 Else
'Compute information gain and normalized information gain and cumulative
information gain. Store all in Info Table.
 Info.Fields(5) = (Uncertain(jnum, i) - Uncertain(jnum, i - 1))
 Info.Fields(9) = (Uncertain(jnum, i) - Uncertain(jnum, i - 1)) / -localMaxE
 Gain(jnum, i) = Gain(jnum, i - 1) + (Uncertain(jnum, i) - Uncertain(jnum, i - 1))
 End If
 Info.Fields(6) = Gain(jnum, i)
 Info.Fields(7) = -Uncertain(jnum, i) / -localMaxE
 Info.Fields(8) = Gain(jnum, i) / -localMaxE
 Info.UPDATE

'Accumulate Total Entropy values over all vehicles and store in Uncertain(red +1,i).
This will be used for Normalization calculations.
 Uncertain(red + 1, i) = Uncertain(red + 1, i) + Uncertain(jnum, i)
'Accumulate Total Gain values over all vehicles and store in Gain(red +1,i). This
will be used for Normalization calculations.
 Gain(red + 1, i) = Gain(red + 1, i) + Gain(jnum, i)
'Get next vehicle.
 MyRecords.MoveNext
 jnum = jnum + 1
 Wend
'Fill the NormEntropy Table with normalized values.
Norm.AddNew
Norm.Fields(0) = i
Norm.Fields(1) = Uncertain(red + 1, i) / MaxE
Norm.Fields(2) = Gain(red + 1, i) / -MaxE
```

```
Norm.UPDATE
Next i

DoCmd.Hourglass False
End Sub
```

| deltaT | Time | PID | STATUS | ENTROPY | Info Gain | Cumulative | normE |
|---|---|---|---|---|---|---|---|
| 3 | 7:23:00 | 1183 | Area | 16.50585 | 0.124971 | 0.2400388 | 0.9856658 |
| 4 | 7:28:00 | 1183 | Area | 16.31249 | 0.1933558 | 0.4333946 | 0.9741194 |
| 5 | 7:33:00 | 1183 | Area | 16.24111 | 7.138275E-02 | 0.5047773 | 0.9698567 |
| 6 | 7:38:00 | 1183 | Area | 16.19769 | 4.342244E-02 | 0.5481998 | 0.9672636 |
| 7 | 7:43:00 | 1183 | Area | 16.15867 | 0.0390198 | 0.5872196 | 0.9649335 |
| 8 | 7:48:00 | 1183 | Area | 16.06786 | 9.080768E-02 | 0.6780273 | 0.9595109 |
| 9 | 7:53:00 | 1183 | Detect | 11.51822 | 4.549641 | 5.227668 | 0.6878238 |
| 10 | 7:58:00 | 1183 | Degrade | 13.50994 | -1.99172 | 3.235948 | 0.8067616 |
| 11 | 8:03:00 | 1183 | Kill | 0 | 13.50994 | 16.74589 | 0 |
| 12 | 8:08:00 | 1183 | Kill | 0 | 0 | 16.74589 | 0 |
| 13 | 8:13:00 | 1183 | Kill | 0 | 0 | 16.74589 | 0 |
| 14 | 8:18:00 | 1183 | Kill | 0 | 0 | 16.74589 | 0 |

Tables 15a and 15b. Example Info table data computed by the ComputeE routine. Shows the status of vehicle 1183 over time.

| normCumGai | normInfo Gain |
|---|---|
| 1.433419E-02 | 7.462784E-03 |
| 2.588066E-02 | 1.154646E-02 |
| 3.014336E-02 | 4.262703E-03 |
| 3.273638E-02 | 2.593021E-03 |
| 0.0350665 | 2.330112E-03 |
| 4.048918E-02 | 5.422686E-03 |
| 0.3121762 | 0.271687 |
| 0.1932384 | -0.1189378 |
| 1 | 0.8067616 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |

Table 15b.

DEPLOYMENT

| DeltaT | StdDev | MoveRate |
|---|---|---|
| 0 | 162.8941 | 0 |
| 1 | 47.58274 | 2.020323 |
| 2 | 28.57213 | 3.220393 |
| 3 | 33.66924 | 8.875961 |
| 4 | 33.84324 | 11.04571 |
| 5 | 36.23609 | 4.049502 |
| 6 | 45.7908 | 2.320711 |
| 7 | 17.35432 | 1.58029 |

Table 16. Vehicle spread as measured by standard deviation of distance from unit center of mass and unit speed determined by magnitude of the movement vector.

```
'Declarations
Option Compare Database
Option Explicit

'This routine uses the global "Variance" array to compute standard deviation of
vehicle spread about the unit center of mass in the normal vector direction. It also
uses the Magnitude Array to compute the unit's movement rate during each time
step.
Sub StdDev()
DoCmd.Hourglass True
Dim MyDb As DATABASE, MyRecords, STD As Recordset
Dim i, v As Integer
Dim strSQL As String, qdf As QueryDef

Set MyDb = DBEngine.Workspaces(0).Databases(0)
'Open the StdDev table.
Set STD = MyDb.OpenRecordset("StdDev_tbl")

'Clear the standard deviation table.
If isTable("StdDev_tbl") Then
strSQL = "Delete*from StdDev_tbl;"
Set qdf = MyDb.CreateQueryDef("", strSQL)
qdf.Execute
End If
'For each time step.
For v = 0 To k
'Write a new record to the StdDev table.
STD.AddNew
   STD.Fields(0) = v 'Time Step
   STD.Fields(1) = Sqr(Variance(v)) 'Standard Deviation
   If Not (v = 0) Then
```

```
'Compute the unit's rate of movement.
      STD.Fields(2) = ((10 * Magnitude(v)) / 1000) / (deltaT / 60)
   End If
   STD.UPDATE

Next v
DoCmd.Hourglass False
End Sub
```

SUPPORTING MINOR FUNCTIONS

```
'Formats the mission number for string comparison i.e.  3 becomes 003 and
'10 becomes 010.
Function mishnum(currentnumber)
If Len(currentnumber) = 2 Then
mishnum = "0" & currentnumber
Else
mishnum = "00" & currentnumber
End If
End Function

'This function computes the distance between two grid coordinates.
Function Distance(reportedloc, actualloc) As Single
Dim xreport, yreport, xactual, yactual, deltax, deltay As Long

xreport = Mid(reportedloc, 1, (Len(reportedloc) - 4))
yreport = Mid(reportedloc, (Len(reportedloc) - 3))
xactual = Mid(actualloc, 1, (Len(actualloc) - 4))
yactual = Mid(actualloc, (Len(actualloc) - 3))
'Since grid coordinates locate down to 10 meters we multiply by 10 to get meters.
deltax = (Abs(xreport - xactual)) * 10
deltay = (Abs(yreport - yactual)) * 10

Distance = Sqr(deltax ^ 2 + deltay ^ 2)

End Function

Function GetTimeStep() As String
GetTimeStep = deltaT
End Function

'This function strips the grid zone designator from a grid.  i.e. MG123456 becomes
123456.
```

```
Function gridparse(gridstr)
gridparse = Mid(gridstr, 3)
End Function


'Returns the Maximum of two numbers.
Function Max(num1, num2) As Single
If num1 >= num2 Then
Max = num1
Else
Max = num2
End If
End Function

'Returns the Minimum of two numbers.
Function Min(num1, num2) As Single
If num1 <= num2 Then
Min = num1
Else
Min = num2
End If
End Function

'Takes a six digit grid and converts to an eight digit grid. i.e. 123456 becomes
12304560.
Function parse_eight(gridstr)
Dim first, second As Integer
If (Len(gridstr) = 6) Then
first = Mid(gridstr, 1, 3)
second = Mid(gridstr, 4)
first = first & "0"
second = second & "0"
parse_eight = first & second
ElseIf (Len(gridstr) = 8) Then
parse_eight = gridstr
Else
MsgBox (gridstr)
End If

End Function

'Converts 090500 to 09:05:00.
Function parse_time(rptime)
If Len(rptime) = 6 Then
parse_time = Mid(rptime, 1, 2) & ":" & Mid(rptime, 3, 2) & ":" & Mid(rptime, 5)
```

```
ElseIf Len(rptime = 5) Then
parse_time = "0" & Mid(rptime, 1, 1) & ":" & Mid(rptime, 2, 2) & ":" & Mid(rptime, 4)
End If

End Function
```

**'Takes 03MAR96 and converts to yymmdd or 960303.**

```
Function parsedate(dte)
Dim first, second, third As String
first = Mid(dte, 1, 2)
second = Mid(dte, 3, 3)
third = Mid(dte, 6, 2)

If second = "MAR" Then
   parsedate = third & "03" & first
ElseIf second = "APR" Then
   parsedate = third & "04" & first
End If

End Function
```

**'Converts 09:00 to 09:00:00.**

```
Function striptime(rptime)
striptime = rptime & ":00"
End Function
End Function
```

**'Returns the time of death of an enemy vehicle.**

```
Function killtime(ID)
Dim MyDb As DATABASE, MyTable As Recordset

Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MyTable = MyDb.OpenRecordset("Kill_tbl", DB_OPEN_TABLE)
MyTable.INDEX = "PrimaryKey"
MyTable.Seek "=", ID

If MyTable.NoMatch Then
   killtime = Null
   Else
   killtime = MyTable![KTIME]
   End If
MyTable.Close
MyDb.Close

End Function
```

```
Function GetNumReport()
GetNumReport = NumReported
End Function
```

**'This function opens the kill table and retrieves a particular vehicle's time of death.**
**'This is called at each time step in the ComputeE routine to see if the vehicle has**
**'been killed yet.**

```
Function killtime(ID)
Dim MyDb As DATABASE, MyTable As Recordset

Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MyTable = MyDb.OpenRecordset("Kill_tbl", DB_OPEN_TABLE)
MyTable.INDEX = "PrimaryKey"
MyTable.Seek "=", ID

If MyTable.NoMatch Then
   killtime = Null
   Else
   killtime = MyTable![KTIME]
   End If
MyTable.Close
MyDb.Close

End Function
```

## REFERENCES


[1] Barr, D., and T. Sherrill. 1996. Measuring Information Gain in Tactical Operations. Operations Research Center Technical Report, U.S. Military Academy, West Point.

[2] Barr, D. and T. Sherrill. 1995. Estimating the Operational Value of Tactical Information. Operations Research Center Technical Report, U.S. Military Academy, West Point.

[3] Barr, D., M. Tillman and S. Strukel. 1994. Entropy Measures of Reconnaissance. Operations Research Center Technical Report, U.S. Military Academy, West Point.

[4] Marin, J. and D. Barr. 1997. Evaluation of Intelligent Minefields. Military Operations Research (to appear).

[5] Shannon, C. 1948. A Mathematical Theory of Communication. The Bell System Technical Journal 27, 379-423.

[6] Sherrill, T. and D. Barr. 1996. Exploring a Relationship Between Tactical Intelligence and Battle Results. Military Operations Research 2, 17-33.

[7] Sherrill, T., M. Johnson, P. West, and D. Barr. 1997. Quantifying Information Gain in Janus. Operations Research Center Technical Report, U.S. Military Academy, West Point.

[8] Willmore, F., D. Barr and D. Voils, *Analytic Geometry, a Vector Approach*, Allyn and Bacon, Inc., Boston, 1971.