2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

MODELING OBSERVATION IN INTELLIGENT AGENTS: KNOWLEDGE AND BELIEF

by

CPT William C. Branley, Jr.

March 1992

Thesis Advisor: Prof. Hemant Bhargava

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b. OFFICE SYMBOL *(If applicable)* 55	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000	7b ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)*	10 SOURCE OF FUNDING NUMBERS: Program Element No; Project No.; Task No; Work Unit Accession Number

11. TITLE *(Include Security Classification)*
MODELING OBSERVATION IN INTELLIGENT AGENTS: KNOWLEDGE AND BELIEF

12. PERSONAL AUTHOR(S) Branley, William C., Jr.

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT *(year, month, day)*	15 PAGE COUNT
Master's Thesis	From To	March 1992	81

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES		
FIELD	GROUP	SUBGROUP

18 SUBJECT TERMS *(continue on reverse if necessary and identify by block number)*
Autonomous agent, uncertain reasoning, degrees of belief, Dempster-Shafer theory, evidential reasoning, expert system, combat modeling, battlefield observation, real-time simulation.

19. ABSTRACT *(continue on reverse if necessary and identify by block number)*

In this paper, a method is presented for controlling autonomous agent behavior by filtering the agent's input. Without such filtering, the agent is allowed to have exact knowledge of the state of its domain, resulting in a pattern of performance that is unrealistic and consistently successful. However, filtering that knowledge into beliefs is a way of making it possible for the agent to be unsuccessful some of the time. That is, if the agent is working from beliefs, and the beliefs happen to be wrong, then the agent may not reach its goal at that particular instant. An application for this method--control of an autonomous combat force in a simulation system--is developed and demonstrated in this paper. The algorithm for generating beliefs about battlefield events models the information gathering system of a combat force. However, this model attempts to simulate the results of the information-gathering system, and not the cognitive or perceptive processes contained in such a system.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS REPORT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Prof. Hemant Bhargava	22b TELEPHONE *(Include Area code)* 408-646-2264	22c OFFICE SYMBOL AS/Bh

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete
SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Modeling Observation in Intelligent Agents: Knowledge and Belief

by

William C. Branley, Jr.
Captain, United States Army
B.S., Columbus College, 1987

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1992

Author [illegible]
William C. Branley, Jr.

Approved by: [illegible]
Prof. Hemant Bhargava, Thesis Advisor

[illegible]
Mr. David Pratt, Second Reader

[illegible]
Prof. David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

In this paper, a method is presented for controlling autonomous agent behavior by filtering the agent's input. Without such filtering, the agent is allowed to have exact knowledge of the state of its domain, resulting in a pattern of performance that is unrealistic and consistently successful. However, filtering that knowledge into beliefs is a way of making it possible for the agent to be unsuccessful some of the time. That is, if the agent is working from beliefs, and the beliefs happen to be wrong, then the agent may not reach its goal at that particular instant. An application for this method--control of an autonomous combat force in a simulation system--is developed and demonstrated in this paper. The algorithm for generating beliefs about battlefield events models the information-gathering system of a combat force. However, this model attempts to simulate the results of the information-gathering system, and not the cognitive or perceptive processes contained in such a system.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. THE AUTONOMOUS AGENT PROBLEM 1

B. NPSNET 1

C. ASSUMPTIONS AND ISSUES 3

D. ORGANIZATION OF THIS PAPER 5

II. KNOWLEDGE AND BELIEF 6

A. INTRODUCTION 6

B. THE OBSERVER ANALOGY 6

C. KNOWLEDGE AND BELIEF 7

D. REASONING WITH UNCERTAINTY 11

E. SUMMARY 12

III. SIMULATING A BATTLEFIELD OBSERVER 14

A. INTRODUCTION 14

B. MODELING BATTLEFIELD CONDITIONS 15

C. THE OBSERVER MODEL 18

D. TARGET CHARACTERISTICS 21

E. DETERMINING THE DISTANCE FACTOR 23

F. THE BELIEF GENERATION ALGORITHMS 26

1. Selecting Beliefs for Continuously-valued Attributes 27

a. Speed 27

b. Direction and View Direction . . . 29
c. Location 30
2. Selecting Beliefs for Discretely-valued Attributes 32
G. SUMMARY 41

IV. AN ILLUSTRATION OF THE MODEL 42
A. INTRODUCTION 42
B. EXAMPLE ONE 43
C. EXAMPLE TWO 46
D. EXAMPLE THREE 50
E. SPEED AND HEADING ERROR 52
1. Speed Error 54
2. Heading and View Direction Error 57
F. SUMMARY 60

V. IMPLEMENTATION 62
A. ORGANIZATION AND SEQUENCE OF EVENTS 62
B. SUMMARY 65

VI. CONCLUSIONS 67
A. A REVIEW OF THE INITIAL ASSUMPTIONS AND STRATEGY 67
B. OVERALL STRENGTHS AND WEAKNESSES 68
C. RECOMMENDATIONS FOR FURTHER DEVELOPMENT . . . 70

D. SUMMARY 70

LIST OF REFERENCES 71

INITIAL DISTRIBUTION LIST 73

I. INTRODUCTION

A. THE AUTONOMOUS AGENT PROBLEM

The word "autonomy" refers to a condition of being independent or self-governing**[Ref. 1]**. In a computer simulation system, an autonomous agent is an object or some other entity that appears to possess sufficient "intelligence" to govern its own behavior. It can execute and monitor its decisions during the course of a simulation. The agent's intelligence comes from a computer program that contains information about some domain or universe of discourse, and a set of rules for using that information.

This thesis is about an autonomous force (AF) application that controls a group of combat vehicles in a simulation system. The vehicles represent a small force of tanks employing battlefield information and background knowledge to make tactical decisions directed toward the accomplishment of an assigned mission.

The remainder of this chapter covers the overall development environment and presents key issues related to the development of autonomous agent applications.

B. NPSNET

The autonomous force program discussed in this paper was developed for the Naval Postgraduate School Networked

Simulator (NPSNET), developed at the Computer Science Department of the Naval Postgraduate School**[Ref. 2]**. The system is a combat simulation environment in which users can interact with each other, as well as with the system. Users of NPSNET may operate any one of several hundred aircraft or ground vehicles. Many of the vehicles are armed, so that users of the simulator can engage each other in mock battles. The battlefield is a 3D representation of a real environment, such as Fort Hunter Liggett, California. It contains mountains, hills, valleys, roads, trees, open spaces and other features. The terrain color can be green to represent vegetation, or brown for generic dirt. The choice of atmospheric conditions ranges from clear to foggy.

The role of the AF program in this simulation is to provide users of NPSNET with an automated opponent. If the simulator were employed as a combat training tool, the availability of an AF program provides many training opportunities, such as being able to fight mock battles on the simulator without the assistance of another user, or forming teams to oppose AF vehicles as a small unit instead of individually.

The employment of autonomous forces has many precedents in the past development of simulation systems. One of the most successful examples is the Semi-automated Forces (SAFOR) used by the U.S. Army's Simulation Network

(SIMNET) **[Ref. 3]**. An important characteristic of the SAFOR approach is the "man-in-the-loop" concept. That is, the automated forces are ultimately controlled by a human decision maker, thus the name 'semi-automated.' In the conduct of a simulation, this setup allows commanders and staffs to control large forces without necessarily using manned simulators. This feature also populates the battlefield with more objects, thus creating a more realistic combat environment.

C. ASSUMPTIONS AND ISSUES

From the beginning, certain assumptions influenced the design and implementation of this AF program. In this section, the key assumption that impacts on the entire organization of the program is discussed.

Each active station in NPSNET can display the current state of the simulation. For example, if a user makes a right turn in a vehicle, the other users will see that vehicle turning right on their screens. Each station relies on state messages from the other stations to keep its display current. A state message is generated by a station whenever its state changes, which can occur, for example, when a user fires a weapon or changes direction.

The first assumption made about the AF program was that it would operate as a station on the network. This meant that the program would routinely process state messages on all other vehicles in the system. Since the AF's job would be to attack and destroy other vehicles in the simulation, it seemed that giving the AF state messages on all other vehicles in the system would give the AF an unfair advantage. It was assumed that the AF, if given perfect information on a vehicle that was designated as a target, would then be able to destroy that target every time.

This latter assumption resulted in an architecture that supported a clear division of responsibilities within the AF program. The AF program would model two basic combat functions: it would model the process of gathering information about events on the battlefield, and it would model the decision-making process that interprets, and acts upon, that information. These two functions formed the basis for two separate research issues associated with the development of this program. This thesis is concerned with the first of those two issues: that of modeling observation or perception on a battlefield. A companion thesis discusses the decision-making function of the program**[Ref. 4]**.

Regarding the research goal of this thesis--modeling battlefield observation--an important distinction must be made. The goal of this research is not to model human perceptive processes, such as vision and hearing. Rather, it is to model the *results* of those processes. For example, instead of modeling how the eye works, the goal is to model what a person might be able to see under certain conditions. As far as the AF is concerned, this means modeling what the AF would be likely to know about its opponents at any given point in time, based on several key factors. This distinction will become clearer in Chapters II and III when the model is presented.

D. ORGANIZATION OF THIS PAPER

In Chapter II the basic strategy for this project is presented, along with an overview of the artificial intelligence issues that impact on the work. Chapter III contains a description of the proposed model for accomplishing the goals stated in the previous section. The model is further explained in Chapter IV with actual output from a series of simulations using the model. In Chapter V, the implementation of the program with its decision-making counterpart is explained. Finally, Chapter VI concludes with a summary of the work and a discussion of the original assumptions and objectives.

II. KNOWLEDGE AND BELIEF

A. INTRODUCTION

The overall approach to this project is based on a real world analogy. It is explained in this chapter, followed by a discussion of the concepts of knowledge and belief that are key features of the adopted strategy. Lastly, the chapter covers reasoning with uncertainty, another issue that is central to the successful implementation of this AF program.

B. THE OBSERVER ANALOGY

As explained in Chapter I, the AF program is divided along two basic functions: battlefield observation and decision-making. Furthermore, the motivation for modeling battlefield observation is to give the AF decision maker information that approximates the information it would be likely to have if it were gathering information on a real battlefield.

On a real battlefield, information is gathered in many ways. For instance, long range patrols roam the forward areas and report by radio. Observation posts hidden on high terrain features scan their assigned sectors and send in reports by radio or field telephone. The commander may also receive information from higher headquarters, neighboring units, and allies. This information may be in the form of photos, descriptions, map overlays, and so on. Finally, the forward elements of the command's fighting force are also information gatherers. Since they are likely to see the enemy first, they can immediately report back on its size, equipment, activities and location.

All of this information is assimilated by the decision-maker and used to form a course of action, or to make changes to a previously chosen course of action. The system, therefore, consists of two subsystems: one that gathers information, and another that uses it. The AF program was structured in the same manner, with roughly the same division of functions. This thesis, in particular, attempts to model the information gathering subsystem by simulating the *reports* that are sent to the decision maker by observers. Again, a distinction alluded to in Chapter I is relevant here: the goal is to model the results of the observation process, not the process itself.

C. KNOWLEDGE AND BELIEF

Recall from Chapter I that, without this observation function, the AF would be free to use NPSNET world state messages to form courses of action. It was assumed that the AF would then have an unrealistic advantage over its opponents. This is analogous to a real world commander who has perfect information about the enemy's size, location, equipment, amount of ammunition, level of fuel in fuel tanks, and so on. Obviously, this is never the case. Even very good information is not perfect. However, if the goal here is to model the results of the observation process, how far from perfect should the information be?

The strategy used here is to distinguish between knowledge, that which is known to be true about something, and beliefs, that which is believed to be true. These concepts have been investigated by many artificial intelligence researchers**[Ref. 5]**, **[Ref. 6]**, **[Ref. 7]**. Davis, for example, describes three degrees of belief possessed by an agent, such as an autonomous agent:

1. Explicit belief -- The program believes anything that is explicitly in the knowledge base.
2. Derivable belief -- The program believes anything that the inference engine can derive in a retrieval.
3. Implicit belief -- the program believes anything that could be inferred in principle via valid inference from the knowledge base.

Implied in the concept of belief is the notion that a belief can be wrong. Therefore, the agent should be allowed to hold beliefs which, although supportable by the knowledge base, are contrary to the true facts about the world.

In the AF program, the agent's knowledge comes from two sources: the static knowledge base that describes certain facts and relationships about the environment, and a dynamic knowledge base that changes with each decision cycle of a simulation. Certain true facts about the world are allowed to be asserted as knowledge, such as the current system time or a vehicle identification number assigned to each vehicle in the simulator by NPSNET. However, most of the remaining facts about the world, such as information describing the targets, are not asserted as knowledge. Instead, they are processed and asserted as beliefs. The observer program then uses knowledge and rules to determine additional beliefs, if they exist. Finally, the beliefs are passed to the decision maker for action.

A critical step, of course, is the part about processing true facts about the world into beliefs. That is the bulk of the work of this thesis. The task is accomplished with several belief generation algorithms that are described in Chapter III and demonstrated in Chapter IV. Placed in the context of the observer analogy, the output of the belief generation algorithms approximates reports that an observer would send under a given set of conditions.

Performing the belief generation task again raises a question introduced in the first paragraph of this section: how far from perfect should the information be? If true facts about something are to be processed into beliefs, how much error should be introduced? More to the point, what is a valid indicator that the belief algorithms successfully model a real battlefield observation?

These questions were resolved in the following manner. Rather than make the assertion "Here is how the average battlefield observer behaves," the assumption is made that battlefield observers in the real world may be highly skilled and very effective, or poorly trained and ineffective, and most of them fall somewhere in between. The model was then designed to accommodate this range of abilities among observers. By adjusting certain parameters and making changes to the knowledge base, the observer can be always right or always wrong, or it can be right or wrong some percentage of the time. With this feature in place, the user can control the performance of the autonomous force. With a poor observer, the AF will be easy to beat because the AF will be working with beliefs that are substantially different from the true facts about the world. But with a good observer, the AF's beliefs may be very close to the truth, and the AF will be harder to beat.

To summarize this section, true facts about the world are converted into beliefs by a set of belief generation algorithms. These algorithms have parameters that can be set to control the amount of error that is introduced. A knowledge base and rule base are used by the program to derive additional beliefs from existing beliefs. The output from the observer subsystem is sent to the decision maker for action.

D. REASONING WITH UNCERTAINTY

The final section of this chapter covers a few reasoning issues that are central to the successful implementation of this model.

The observer program discussed in the previous section can be thought of as an expert system that predicts what a human observer's report would be under a given set of circumstances. At any given instant in time, the system must choose a belief about something from among several possible beliefs, all of which may be equally valid. For example, if the true state of a vehicle is that it is traveling at 32 kilometers per hour, then when is a belief of 30 kph more appropriate than a belief of 28 kph? An additional problem with modeling a real world observer is that no one, not even the observer, can predict with *certainty* what his or her next report will be.

This is a standard problem that has been characterized by many researchers as reasoning with uncertainty **[Ref. 8]**, **[Ref. 9]**, **[Ref. 10]**. In general, reasoning with uncertainty means having to form a conclusion about something based on incomplete or incorrect information. In some applications, such as medical diagnosis, the system must undertake a complex reasoning process to arrive at a conclusion that may only have a certain probability of being right **[Ref. 11]**.

The AF application is not as complex as medical diagnosis, but it still requires the use of uncertain reasoning techniques to resolve ambiguities. For example, what would an observer really report? And what would constitute a realistic approximation of that report?

The strategy adopted in this application uses several standard techniques for dealing with these issues. However, they are best explained in the context of the model. Therefore, further explanation is reserved for the next chapter.

E. SUMMARY

This chapter developed the analogy of the combat observer sending reports to a decision maker as the basis for the division of functions within the overall AF program. It further portrayed the observation function as a model of the combat observation subsystem in the real world. However, the distinction was made that this program attempts to model the results of the observation process, not the process itself.

This led to a discussion of the basic strategy used by this program to accomplish its goal: the conversion of true facts about the world state into beliefs using a set of belief generation algorithms, a knowledge base, and a set of rules. This process is the heart of this entire program. Next, the chapter covered several related issues concerning the concepts of knowledge and belief, and the use of uncertain reasoning techniques.

In the next chapter, the model and its parameters are laid out in detail. Chapter IV contains a demonstration of its implementation on NPSNET.

III. SIMULATING A BATTLEFIELD OBSERVER

A. INTRODUCTION

In this chapter the mechanism for converting true facts about the world into beliefs is explained in detail. It begins with a discussion of conditions on a battlefield that affect a human observer's performance, as well as individual aspects of the observer that affect how well he or she would perform under any circumstances. The chapter then covers the manner in which the battlefield conditions are combined into a formula that captures the overall effect of the conditions upon the belief generation process. With this formula in place, the functions that compute the actual beliefs can be shown.

Also covered in this chapter are the basic characteristics of objects in NPSNET. These characteristics constitute the true facts about the world that this program converts into beliefs. Finally, the inference procedures that derive new beliefs based on the results of the algorithms are covered in detail. There is also a feature for determining new beliefs based on beliefs from a previous decision cycle.

B. MODELING BATTLEFIELD CONDITIONS

A total of five battlefield conditions are simulated in this program. They are distance, visibility, judgement, knowledge, and equipment. Each of them is explained in this section. Basically, distance and visibility are meant to represent the physical characteristics of the combat environment, while knowledge and judgement apply to the individual observer. Equipment is related to the observer's capabilities since it denotes any items of equipment used to aid the observation process.

In theory, almost anything can be regarded as an important battlefield factor: the amount of time since the observer's last meal, the amount of sleep in the prior 24 hours, the actions of the objects that are being observed, knowledge about the combat environment in which they are operating, and so on. Since the automated force program must operate in real time, it is not desirable to try to model *every* condition under which people make judgements on a battlefield.

Distance. Denoted by *d*, it is a number between 0 and 1 that is based on the actual distance between the observer and the target. A value of 1 means the object is so close that it can be positively identified with the naked eye, a value of 0 means that there is no chance that the attribute can be identified. Choosing to model this battlefield condition rests on an assumption that distant objects are much harder to identify than relatively close objects. In Section E, a formula will be presented that shows how *d* is computed, given the actual distance to the target.

Visibility. Denoted by *v*, it is a value between 0 and 1 that describes atmospheric conditions, such as darkness, fog, smoke, dust. It is important to note that this factor covers anything that impacts on visibility, to include time of day. A value of 0 for *v* denotes the poorest conditions, such as a moonless night or impenetrable fog, when visibility may be less than 30 feet. $v = 1$ means perfect viewing conditions, when visibility may be 5,000 meters or more with the naked eye. Values between 0 and 1 denote various visibility ranges. By default, a simulation begins with a visibility of 1, assuming a daylight battle in a clear desert environment. However, the presence of smoke, dust, and haze can change that value during a simulation. Also, a user of NPSNET can change the visibility via menu selection during a simulation.

Judgement. Denoted by *j*, a value between 0 and 1 is used to represent the skill and experience of the observer. This can include the observer's eyesight, alertness, intelligence, enthusiasm about the mission, and so on. The number is subjectively set, with 1 representing a highly-skilled observer, and 0 meaning the opposite. Although judgement is an important individual characteristic, it does not influence AF performance as much as some other characteristics, such as knowledge or equipment. The default value is $j = 1.0$.

Knowledge. Denoted by k, this value also falls between 0 and 1. It is a measure of what the observer knows, through prior knowledge or external sources, about the objects being observed. If knowledge is 1, then the true facts are always known to the observer, and are asserted as beliefs with a value of 1, which has the same effect as asserting them as knowledge. A value of 0 means that the observer has no relevant knowledge about the objects it sees. The default value is 0.5.

Equipment. Denoted by q, this is a measure of the utility of various pieces of equipment available to the observer for viewing the battlefield, such as binoculars, night vision devices, and so on. The naked eye is denoted by a value of 0, and 1 means that the equipment used provides maximum utility. By default, the observer has binoculars and night vision equipment. The value of q changes during the simulation as the observer uses different equipment in different situations. If desired, the user can take away the equipment at startup, forcing the observer to use only the naked eye.

The factors described above are designed to represent only the most fundamental battlefield conditions. Notice that noise is not accounted for, even though battlefield sounds provide significant clues to identifying things. However, noise and other factors will be discussed in the final chapter as issues for future program development.

C. THE OBSERVER MODEL

This section explains how the five battlefield conditions, Table I, are combined into a single parameter that represents the overall effect of those conditions upon the belief generator's output.

Table I: BATTLEFIELD FACTORS

	Range	
Factor	Best	Worst
d	1	0
v	1	0
j	1	0
k	1	0
q	1	0

A function, denoted as m, represents the total effect of d, v, j, k, and q. $m(d,v,j,k,q)$ is interpreted as a probabilistic measure of the ability of the observer to make a correct observation under the specified conditions. *m* itself must fall between 0 and 1, since it is a measure of probability. If *m* is low, then there is relatively little chance that the observer will observe and correctly identify selected attributes of an object. If *m* is high, then there is a relatively greater chance that the observer's report will be accurate. Output generated by the observer module becomes input to the decision maker. This means that when *m* is low, the decision maker will probably be working with poor information. However, it will be shown that *m* rises as the AF gets closer to its target, so the decision maker gets better and better information with each decision cycle.

Since the probability function *m* is somewhat unwieldy in its entirety, some of the factors are combined into intermediate functions denoted by the letters *c* and *g*. The functions are shown here and then explained in turn.

$$c(d,v,j)=d^2e^{-\lambda(1-j)}v \qquad \lambda = 0.5 \qquad \textbf{(1)}$$

$$g(c,q)=c+q(1-c)^{2-q} \qquad \textbf{(2)}$$

$$m(g,k)=g^{1-k} \qquad \textbf{(3)}$$

Equation (1) depicts a multiplicative relationship between *d*, *v*, and *j*. In order to have $c = 1$, all three must equal 1. The desire here was for these three parameters to have a strong combined effect on m, but that one of them alone could not force m to equal 1. This is based on an intuitive interpretation of these parameters. For instance, judgement alone should not result in perfect knowledge, clear visibility alone should not lead to a perfect observation, and close proximity should not automatically result in a perfect observation because the observer can make a mistake even at close range. However, the combined effect of these three parameters, which is c, will always force m to 1.

The next two functions will be discussed together. In function $g(c,q)$, Equation (2), the relationship between c and q is such that if either of these is 1, then $g = 1$. Then, in $m(g,k)$, the same relationship holds for g and k. The overall effect is that either knowledge, or equipment, or the combined effects of distance, visibility, and judgement can produce the result $m = 1$. This supports the goal that m should be 1 when knowledge is 1, and it also supports the intuitive notion that the use of equipment has a dramatic effect on the abilities of the observer. For example, using binoculars enables observation of distant objects with much more clarity than is possible with the naked eye.

D. TARGET CHARACTERISTICS

In this section, the key characteristics of NPSNET vehicles are discussed. The values of these characteristics come to the AF program via world state messages and are ultimately converted to beliefs by the battlefield observation program.

Table II, next page, shows the contents of a state message. These messages are generated on startup and when there are changes to predetermined aspects of a vehicle's state, such as its speed and direction. Between state messages, the AF program must apply its own dead reckoning algorithm to keep track of the locations of vehicles in the simulation.

As stated above some items of information from the state message are asserted as facts without belief processing; namely, the system time and the vehicle ID. In addition, certain other items are ignored, such as the elevation of the target's gun and chassis. For one thing, it is not likely that these would ever be known by an observer. Also, a combat force does not have a need for such detailed information about its opponents.

Table II: CONTENT OF STATE MESSAGE

Field Name	**Description**	**Values**
int hours	time of message	
int minutes	"	
int seconds	"	
int mills	"	
int vehno	unique ID for each object in the simulation	
int vehtype	index to an array containing a description of the vehicle	
int gunfire		0 = has not fired 1 = has fired
int alive		0 = dead 1 = alive

float pos[3]	location of vehicle in 3D world coordinates, using the right-handed coordinate system	x = east-west axis
		y = vertical axis, or elevation
		z = north-south axis

float direction	the direction the vehicle is heading	0 - 359 compass degrees
float viewdirection	the direction in which the vehicle is looking, or the turret state	0 - 359 compass degrees, relative to direction
float elev	elevation of chassis	
float gunelev	elevation of main gun, relative to chassis	
float speed	speed in kilometers	0 to max speed

The remaining elements of the state message are divided into two groups: those that are continuously-valued in nature, and those that are discretely-valued. A continuously-valued element, or target attribute, is one that has a range of allowable values. These are *location, speed, direction*, and *view direction*. A discretely-valued attribute is one that has only a few allowable values. This list includes *vehicle type, gunfire*, and *alive*. The need for this division of target characteristics will become apparent later in this chapter.

Finally two other discretely-valued attributes about a target are considered: *armament* and *armor-protection*. These are not contained in the state message but are implied by the vehicle type of the target. For example, if the target is a tank, then it will have a main-gun and it will be an armored vehicle. These additional target attributes are used in the inference process that determines beliefs about discretely-valued attributes.

E. DETERMINING THE DISTANCE FACTOR

This section is devoted to an explanation of the formula used to compute the distance parameter, which is one of the five battlefield factors. The computation of *d* is unique in that it is not the same for all target characteristics.

First of all, an assumption is made that some target attributes are harder to identify than others. For example, at a range of 4,000 meters, determining which way the turret of a tank is facing is harder than judging the general direction in which the tank is heading. Or, if a vehicle is stationary, it might be hard to tell if it is alive or dead. Therefore, it was desirable that *m* should be given the chance to assume different values when generating beliefs about different target attributes. To accomplish this goal, the formula for computing *d* is given inputs that may vary according to the target attribute that is being evaluated.

As noted earlier, *d* is a number between 0 and 1, and is based on the distance from the observer to the target. When $d = 1$, it means that *an attribute* of a target is so close that it can be positively identified. For each attribute, a value, d_1, represents the range, in meters, at which that attribute can be positively identified. d_0 represents the range at which the attribute can just begin to be seen, but is not likely to be identified accurately. Table III lists these distance values for the various target attributes.

Table III: DISTANCE VALUES, IN METERS.

Attribute	d_1	d_0
vehtype	600	5000
arms	500	3000
armor	550	5000
control	500	4000
gunfire	800	6000
alive	500	5000
location	700	8000
speed	1000	8000
direction	1000	8000
viewdirection	400	4000

To obtain *d* for an attribute of interest, the values for d_0 and d_1 for that attribute are entered into (4) below.

$$d = 1 - \frac{d_a - d_1(attribute)}{d_0(attribute) - d_1(attribute)} \qquad \textbf{(4)}$$

where d_a is the actual distance from the observer to the target. To illustrate, if an object is 2,000 meters away, and the observer is trying to determine if it is an armored vehicle, as opposed to a truck, then *d* would have a value of,

$$\frac{d_a - d_1(armor)}{d_0(armor) - d_1(armor)} = \frac{2000 - 550}{5000 - 550} = 0.33$$

but if the observer were trying to judge the direction in which the vehicle was heading, then *d* would be,

$$\frac{d_a - d_1(direction)}{d_0(direction) - d_1(direction)} = \frac{2000 - 1000}{8000 - 1000} = 0.14$$

If the other factors are held constant, then the difference in *d* for the two examples above will result in a slightly higher *m* in the second case. (Remember, higher values of *d* are more favorable.) This means that the observer has a slightly *better* chance of judging a vehicle's direction of travel than of determining whether or not the vehicle is armored.

F. THE BELIEF GENERATION ALGORITHMS

This section covers the procedures that convert true facts about the world state into beliefs. Beliefs for most of the target attributes are computed by a single function that receives as inputs a true state value for an attribute and the *m* value for that attribute. Some of the target attributes are more complex, requiring a series of functions and rules. For each target attribute there is also a procedure for determining a belief based on a previous belief. This supports the idea that old beliefs influence new beliefs.

In the first part of this section, the belief generation algorithms for *location, speed, direction* and *view direction* will be presented. Following that are the procedures for determining beliefs about *vehicle type*, *armament*, *armor-protection*, *gunfire*, and *alive*.

1. Selecting Beliefs for Continuously-valued Attributes

Beliefs in this category are computed with probabilistic algorithms that introduce error directly into the true value to obtain a belief. Since the value, m_a, is interpreted as the probability of an observer correctly identifying attribute a, then $1 - m_a$ is the chance of error in the observation. For instance, if $m = 0.85$, then there could be as much as a 15 percent error in the observation. The question is, 15 percent of what? For each attribute the error is introduced in a slightly different way, but in each case the result is a random selection based on m. The following subsections will cover the procedure for each of the attributes in the continuously valued category.

a. *Speed*

The arguments to the speed error function are m_{speed} and the actual speed of the target. First, $1 - m$ is used to determine an interval around the actual speed value. For example, if $1 - m = 0.2$, and the true speed is 25 kilometers per hour, then the resulting interval will be as follows

$$lb = 25 - 0.2(25) = 20 \text{ kph}$$
$$ub = 25 + 0.2(25) = 30 \text{ kph}$$

A value is then selected at random from this interval. This scheme has the desired characteristic that the true speed always has a chance of being selected, and it has a greater chance as *m* increases. Also, when the target is moving faster, a given percentage of error will be greater since the percentage is applied to a higher value.

The method just described is used to calculate the initial belief in speed. However, there is a chance that an observer's initial belief about an observation will influence later beliefs. To accommodate this possibility, subsequent speed belief calculations use a function that receives as arguments the actual speed and the previous speed belief. The interval from which the random selection is made is the interval formed by taking the absolute value of the current true speed subtracted from the last belief. This means that when a target's speed is constant, the successive beliefs stand a good chance of being closer and closer to the actual speed. This characteristic of the algorithm agrees with the notion that, if the observer views an object moving at a constant speed for a long time (perhaps several seconds), then the observer will be more likely to figure out, or guess, the correct speed.

b. Direction and View Direction

Since these two attribute beliefs are computed in an identical manner, they will be presented together. The arguments to the compass error function are *m* and the true direction in which the target is traveling. A compass error in percentage terms is interpreted as a percentage of the entire compass, i.e. 360 degrees. Therefore the error, $1 - m$, is converted to an error in degrees as follows:

$$error = 360(1 - m)$$

The error is then added to, and subtracted from, the true error to form an interval from which a value is selected. This value becomes the initial belief about direction. Subsequent beliefs are selected at random from the interval formed by taking the difference between the current true value and the most recent belief. For example, if the true value is 255 degrees at time t_1, and the belief at time t_0 was 315 degrees, then the interval from which the new belief is selected would be $315 - 255 = 60$. Note that there can be two interpretations of the difference between two directions on a compass. For example, the difference between 315 degrees and 255 degrees could also be 300. In this application, the smaller difference is always chosen.

This algorithm again has the characteristic that, if direction remains constant for a period of time, the beliefs will gravitate toward the true value. The same arguments discussed above apply here.

c. Location

The location error algorithm is the most complex, and the most important, of the continuously valued belief generation algorithms. Of all target characteristics, the location of the target is the most critical as far as the AF is concerned. The true location of a target must eventually be known if the AF is ever to successfully attack that target. This is different from a real world combat system, where a tank gunner, for instance, can hit a target by aiming at it. The gunner does not need to know the grid coordinates of the enemy tank. The AF, of course, does not have a human aiming mechanism. Its "cross-hairs," so to speak, are the grid coordinates of the target.

For these reasons, the generation of location beliefs is really an attempt to model the effectiveness of a tank gunner. If a gunner is 90 percent effective, then it means that he hits his target 90 percent of the time. In the AF program, this could be simulated by giving the AF the correct grid coordinates 90 percent of the time. Of course, the AF has other internal criteria that must be met before it can hit a target, such as range and line-of-sight. However, if these criteria are satisfied, and the AF is given the correct location of a target (and all of this information is current)[1], it will always hit the target.

The location belief algorithm is designed to model gunner effectiveness by giving the AF the correct location a certain percentage of the time. Its arguments are $m_{location}$, the true X coordinate and the true Z coordinate. (Note: The Y coordinate denotes elevation in the NPSNET world coordinate system.) The function returns a set of believed X and Z coordinates.

The basis for the error is $1 - m$. Error in location is interpreted as a percentage of the distance between the AF and the target. If the target is 1000 meters from the AF, and $1 - m = 0.05$, then

$$error = 0.05(1000) = 50 \text{ meters}$$

A random number selected from the interval [0, 50] would become the actual error contained in the belief. However, there is an additional step. In order to compute X and Z coordinates for the believed location, a direction from the target is chosen at random and then used, along with the distance, to determine the believed coordinates.

[1]. The information may not be current if the network becomes overloaded. This is an implementation problem that will be discussed in Chapter V.

The direction to the target is only chosen at random during the initial belief calculation. So that a series of beliefs are consistent with each other, subsequent beliefs are chosen, randomly, from an area that is in the same general direction as the initial belief. The distance factor for subsequent beliefs is chosen from an interval formed by taking the difference between the current location and the previous belief. As with the other belief algorithms, subsequent beliefs gravitate toward the true belief as long as the location is unchanged.

This section covered the procedures for determining beliefs about the four continuously-valued attributes. In each case, a percentage of error was applied in some manner to the true value of the attribute. The actual result was a random choice.

2. Selecting Beliefs for Discretely-valued Attributes

This section is concerned with the *vehicle type, gunfire*, and *alive* target attributes that are elements of the NPSNET world state message. The alive attribute is important to the AF because if a target is dead, it should be withdrawn from the target list. The gunfire attribute is useful, but not critical at this stage of AF development because the AF does not attempt to "dodge" incoming rounds or take cover. Finally, the vehicle type attribute is another item for future development of the program. At present, the AF will attack any vehicle that is asserted as a target. However, some targets, such as a tank, should be given greater priority than others, such as a truck. Also, if the AF is to follow international protocols, then it should not attack ambulances or other medical aid vehicles.

Since these features are all possibilities for future development of the AF, they have been included in the belief generation program and will be discussed in this section. Although these procedures have been implemented in the program, they can be effectively turned off to save computation time.

The procedure for generating discretely-valued attribute beliefs is a multi-step process that uses several functions, rules, and facts from the knowledge base. The basic strategy is to compute intermediate beliefs for one or two selected attributes and then treat the intermediate beliefs as evidence to support other beliefs. To support this strategy, it was necessary to designate additional target characteristics that could be used for generating intermediate beliefs. The attributes, *armament* and *armor protection*, were chosen for this purpose. Although they are not included in the NPSNET world state message, they are useful for making inferences about the vehicle type and gunfire attributes.

These special target attributes have values that are associated with one or more categories or families of vehicles. For example, all of the varieties of tanks modeled in NPSNET belong to the family of tanks. Armored vehicles that carry troops to combat belong to the armored personnel carrier family, and so on. This data, provided in Table Table IV represents the *armament* and *armor protection* attributes.

Table IV: INTERMEDIATE ATTRIBUTE VALUES

Attribute: Armament

Value	Vehicle Type Category
main-gun	tank
small-arms	apc (armored personnel carrier)
anti-armor	itv (improved TOW vehicle)
arms-none	truck

Attribute: Armor Protection

Value	Vehicle Type Category
has-armor	tank, apc, itv
no-armor	truck

When an NPSNET world state message is received by the observer program, the value of the *vehicle type* attribute is checked against lists of vehicle types stored in the observer's knowledge base. If the vehicle is a tank, then the facts about the tank's armament and armor protection will be asserted. Namely, that the vehicle has a main-gun and it is an armored vehicle. These new facts are then treated as true world state facts and are subject to belief computations.

Because of the nature of discretely-valued attributes, a selection method like the one described in Subsection 1 is not appropriate. For one thing, if an attribute has only two or three allowable values, selecting a belief by computing a percentage error based on the true value is inappropriate. Also, the method described in the previous subsection does not exploit the fact that some attributes lend themselves to inference. Instead of using the methods described previously, an evidential reasoning technique based on Dempster-Shafer theory is used to take advantage of the fact that discrete attributes can be both observed and inferred**[Ref. 12],[Ref. 13]**.
Dempster-Shafer theory is a mathematical, non-Bayesian theory for combining multiple pieces of evidence to develop beliefs in various hypotheses. It features a tableau mechanism that is especially suited to combining the types of evidence considered by this belief generation procedure. To illustrate the use of this technique, and the overall procedure, the next several paragraphs will be devoted to generating a belief about vehicle type.

The first step is to calculate *m* for all discretely valued attributes, to include the intermediate attributes. Each *m* is then evaluated as evidence supporting the actual value of the attribute, and $1 - m$ is assigned to the complement of the true value. For example, if armor protection is being considered, and $m_{armor} = 0.75$, then the following evidence would be generated and labeled m_1:

m_1({armor:has-armor}) = 0.75
m_1({armor:no-armor}) = 0.25

Likewise, if the vehicle type is tank, and $m_{vehtype} = 0.85$, then:

m_1({vehtype:tank}) = 0.85
m_1({vehtype:apc,itv,truck}) = 0.15

For each attribute, m_1 is the label assigned to evidence based on direct observation of the attribute, since it is derived from *m*, and *m* itself is the probability that an attribute was observed and identified correctly.

After m_1 has been determined for each attribute, certain rules are activated that try to apply facts in the knowledge base to existing evidence. For example, two of the facts are:

```
(has-armor (tank apc itv) 1.0)
(no-armor (truck) 1.0),
```

where the numerical value, called *p*, is the probability that the first term in the fact implies the second term. These particular facts say that the presence of armor plating on a target implies that the target is either a tank, or an apc, or an itv, with *p = 1*, and the absence of armor implies that the target is a truck, also with *p = 1*. This may seem redundant, since the associations between these attributes was just given. Remember, however, that the original data was asserted as true facts, and now the program is computing beliefs.

When facts in the knowledge base are matched with existing evidence, new evidence is asserted that contains a numerical value found by multiplying m_1 with *p*. Using the armor protection implications given above, the new facts would be:

```
(belief (has-armor) implies (tank apc itv) 0.75)
(belief (no-armor) implies (truck) 0.25)
```

These new implication facts trigger another set of rules that gather all updated facts about the target attributes and labels that as evidence m_2. m_2 evidence is any evidence in support of a particular belief that is inferred by the presence of another belief. Using the examples thus far, m_2 evidence would include evidence that says something about vehicle type because of the beliefs in armor protection shown above. This is an illustration of how an intermediate belief is used in the belief generation process. Recall that the purpose of this example is to generate a belief in vehicle type. Thus far, the list of evidence about vehicle type consists of:

m_1({vehtype:tank}) = 0.85
m_1({vehtype:apc,itv,truck}) = 0.15
m_2({vehtype:tank,apc,itv}) = 0.75
m_2({vehtype:truck}) = 0.25

The next step is to combine this evidence to form a belief about vehicle type. An intersection tableau is a convenient tool for this. Figure 1 is a simple graph where the evidence to be combined is placed in the margins. For this example, the left margin contains the m_1 evidence and the top margin contains m_2.

	{tank,apc,itv} 0.75	{truck} 0.25
{tank} 0.85	{tank} 0.6375	{} 0.2125
{apc,itv,truck} 0.15	(apc,itv} 0.1125	{truck} 0.0375

Figure 1: Use of intersection tableau to combine evidence values.

The data inside the tableau consist of the intersections of the sets and the product of the values associated with each set. for example, {tank} 0.6375 was obtained by taking the intersection of {tank} and {tank,apc,itv}, and the product 0.75 x 0.85.

The values are then normalized by first totaling the empty sets and subtracting that amount from 1. In the notation provided by the Dempster-Shafer model, this would be:

$$m_1 \times m_2(\{\}) = 0.2125$$
$$\kappa = 1 - m_1 \times m_2(\{\}) = 0.7875$$

The second step in normalization is to divide the values of the non-empty sets in the tableau by κ :

$$m_1 \oplus m_2(\{\text{tank}\}) = 0.6375/0.7875 = 0.8095$$
$$m_1 \oplus m_2(\{\text{apc,itv}\}) = 0.1125/0.7875 = 0.1428$$
$$m_1 \oplus m_2(\{\text{truck}\}) = 0.0375/0.7875 = 0.0476$$

The final step in the belief generation procedure is to treat the results of the intersection tableau as weights in a random selection process. In this case, a number chosen at random from 1 to 100 would be compared to each of the tableau results. If the numb[illegible] 81 or less, then tank would be chosen as the belief for vehicle type. If the number is between 82 and 96, then either apc or itv would be chosen (at random); and if the original random number was between 97 and 100, truck would be chosen as the belief. One of the advantages of this belief generation procedure is that it uses a method of logical inference to determine weights, but then provides for human error by allowing random selection. In the example above, the observer could mistakenly call the tank a truck.

The remaining discretely valued attributes--gunfire and alive--are also computed using a combination of functions, rules and random selection. All of the discretely valued attributes have many inter-relationships. For example, if the value of gunfire is "firing," then that implies something about armament, armor protection, vehicle type, and alive. Although only one example was shown here, the program goes through this process for all of the attributes in this category.

A controlled degree of random selection is important when determining these beliefs because it allows the unpredictable to occur. In a combat environment, for example, people often misidentify things that are spotted on the battlefield. One of the goals in this project was to simulate logical errors, such as mistaking an apc for a tank, as well as judgmental errors that occur in the battlefield observation process.

G. SUMMARY

This chapter covered the functions and procedures that model the battlefield observation process. It described the battlefield conditions that are combined into a value called *m* that is a probabilistic measure of the accuracy of an observation. It then covered the ways that *m* is used to calculate continuously valued and discretely valued target attribute beliefs.

In the next chapter, an analysis of actual data generated by the continuously-valued belief algorithms is presented and discussed.

IV. AN ILLUSTRATION OF THE MODEL

A. INTRODUCTION

In this chapter, several detailed examples are used to show how the belief algorithms work in an actual simulation. Since continuously valued attributes have the greatest impact on autonomous force (AF) behavior, they will be the focus of this chapter.

The first three examples are devoted to explaining the location belief algorithm, which is the most critical belief that is generated by the program. That is because, to the AF, the location of a target is the most important piece of information needed to conduct a successful attack. The last two examples in the chapter are devoted to belief algorithms for speed, heading and view direction.

The examples are presented as a series of engagements pitting an AF of four tanks against various targets in the simulator. Output from the simulations was captured in a file and is used here to show the workings of the model. Before describing the engagements, a word about the targets is necessary.

Vehicles and other moving objects in NPSNET are controlled in one of four ways: scripted, driven, uncontrolled, and autonomous. When the simulator is started, the "world" is initially populated with an assortment of scripted and uncontrolled vehicles. The user can "drive" a vehicle by selecting a two-dimensional vehicle icon with a mouse pointer. The vehicle selected then becomes a driven vehicle. At any time, the user may start the AF program. As far as the AF is concerned, any driven vehicle is a target. Scripted, uncontrolled, and other autonomous vehicles are ignored, with one key exception. If a user leaves a driven vehicle to drive a new vehicle, the previous vehicle becomes uncontrolled, that is, the settings chosen by the driver, such as speed and direction, remain in effect. However, the AF platoon will continue to treat *that* uncontrolled vehicle as a target. Once a vehicle becomes a target, it remains a target until it is either destroyed or it moves out of range.

B. EXAMPLE ONE

Most of the "battles" in this series of examples occur on flat terrain characterized by good line-of-sight for distances of several thousand meters. Under these conditions, the AF's requirements for attacking a target--range of 5,000 meters or less and no obstacles--will be easily met.

As explained in Chapter III, the five parameters used by the belief algorithm are combined into a value labeled *m* that represents the probability that an observation is correct, given the conditions under which the observation is made. Also, *1 - m* is the amount of error that could be present. In other words, if there is a 98 percent chance that an observation is correct, then there may be as much as a two percent error in the observation. When *m* is 1.0, then the observation is perfect, which means the belief matches the actual data. Three things can cause *m* to equal 1.0:

1. If knowledge (*k*) = 1.0, then *m* = 1.0
2. If equipment (*q*) = 1.0, then *m* = 1.0
3. If judgement (*j*) and visibility (*v*) and distance (*d*) all equal 1.0, then *m* = 1.0

In Figure 2, the parameters have been set so that belief generation is effectively turned off. That is, *m* is always 1.0, thus illustrating how the AF performs when it is passed perfect information.

KNOWLEDGE TO BELIEF CONVERSION

No distortion of facts.

k, v, j, q = 1.0

Line				Actual Data			Beliefs		
Number	Veh ID	Time	Type	Range	Xcoord	Zcoord	Xcoord	Zcoord	*m*
1.	31	177006.58	52	1841.65	35843.35	25290.90	35843.35	25290.90	1.00
2.	31	177011.59	52	1799.80	35850.96	25298.42	35850.96	25298.42	1.00
3.	31	177016.27	52	1752.89	35862.90	25310.87	35862.90	25310.87	1.00
4.	31	177020.87	152	1713.22	35867.81	25317.81	35867.81	25317.81	1.00

Figure 2: Data from Example 1 of simulation.

The following column descriptions are provided:

1. Line Number -- The line number of the output.

2. Veh ID -- This is a unique number assigned by NPSNET to each vehicle that is present in a given simulation. When a vehicle is destroyed, its number remains unused for the rest of the simulation.

3. Time -- The system time, useful for measuring elapsed time from the start of an engagement.

4. Type -- The type of vehicle, such as tank, truck, etc.

5. Range -- Range in meters from the attacking platoon to the target.

6. Actual Data -- True location of the target in the NPSNET world coordinate system, in which X represents width, Z represents depth, and Y represents height. Only X and Z are shown here.

7. Beliefs -- The believed location of the target.

8. *m* -- The probability that a belief matches the truth. $1 - m$ is the amount of error that may be introduced.

The first engagement begins with the AF attacking Vehicle Number 31 from a range of 1841.65 meters. Notice that $m = 1.0$. As expected, X and Z coordinates in the Beliefs column are identical to those in the Actual Data column. The type of target is denoted by the number 52, which, in NPSNET, is an armored vehicle called a BMP. However, the fourth line of output lists the vehicle type as 152. This is simply NPSNET's symbol for a destroyed BMP. By convention, the graphical picture of a destroyed vehicle is denoted with a number that is easily related to the type designation of its live counterpart, such as 52 and 152. The point here is that the presence of type 152 in the fourth line means that Vehicle Number 31 is destroyed as of that point in the program.

The elapsed time was about 14 seconds from the start of the engagement until the target was destroyed. It was destroyed at a range of 1713.22 meters. Not shown in the output is the speed of the target, which was under five kilometers per hour. The target was also shooting at, and missing, the AF platoon.

Under these circumstances, it is very difficult for a human opponent to survive an engagement with a AF platoon before being destroyed. The next example shows how a small degree of knowledge-to-belief conversion can give an opponent more time to respond to the AF attack.

C. EXAMPLE TWO

The parameter settings and data for Example 2 are shown in Figure 3. In this and the following example the format of the output has been changed to emphasize the important information. The column labeled Error is the difference, in meters, between the target's true location and its believed location. The next column, %Error, is the amount of actual error in percentage terms. This is obtained by dividing Error by Range. This simulation begins at a range of 4307.49 meters, Line Number 1, at which point $m = 0.76$, thus introducing as much as a 24 percent error. Actual error was

```
              KNOWLEDGE TO BELIEF CONVERSION

        Introducing location error into the SAF input.

            k = 0.8    v = 1.0    j = 1.0    q = 0.0


 Line
Number  Veh ID   Time      Type    Range     Error   %Error    m

  1.      31   697819.81    52    4307.49   646.12    0.15   0.76
  2.      31   697826.37    52    4276.28   983.54    0.23   0.76
  3.      31   697832.94    52    4241.68   424.17    0.10   0.77
  4.      31   697840.56    52    4196.54   713.41    0.17   0.77
  5.      31   697847.25    52    4152.86   705.99    0.17   0.77
  6.      31   697855.69    52    4094.70   286.63    0.07   0.78
  7.      31   697862.75    52    4043.01   80.86     0.02   0.78
  8.      31   697869.75    52    3959.66   277.18    0.07   0.79
  9.      31   697876.75    52    3876.25   697.73    0.18   0.80
 10.      31   697883.69    52    3781.48   37.81     0.01   0.80
 11.      31   697890.87    52    3695.81   110.87    0.03   0.81
 12.      31   697898.25    52    3607.91   216.47    0.06   0.82
 13.      31   697905.25    52    3512.23   210.73    0.06   0.82
 14.      31   697912.37    52    3405.96   204.36    0.06   0.83
 15.      31   697920.06    52    3291.32   526.61    0.16   0.84
 16.      31   697927.81    52    3199.02   319.90    0.10   0.85
 17.      31   697935.69    52    3091.49   0.00      0.00   0.85
 18.      31   697945.06    52    2979.85   89.40     0.03   0.86
 19.      31   697954.25    52    2870.44   373.16    0.13   0.87
 20.      31   697963.19    52    2758.47   165.51    0.06   0.88
 21.      31   697972.06    52    2639.45   158.37    0.06   0.88
 22.      31   697982.94    52    2501.11   275.12    0.11   0.89
 23.      31   697995.44    52    2370.35   71.11     0.03   0.90
 24.      31   698009.12    52    2180.75   152.65    0.07   0.91
 25.      31   698020.06    52    2029.27   20.29     0.01   0.92
 26.      31   698034.00    52    1836.28   73.45     0.04   0.93
 27.      31   698042.37    52    1683.18   100.99    0.06   0.94
 28.      31   698052.00    52    1549.43    0.00     0.00   0.95
 29.      31   698060.62    52    1455.46   43.66     0.03   0.96
 30.      31   698069.12    52    1347.98   40.44     0.03   0.96
 31.      31   698078.50    52    1213.91   36.42     0.03   0.97
 32.      31   698088.25    52    1087.70   21.75     0.02   0.98
 33.      31   698097.62   152    966.37    9.66      0.01   0.99
```

Figure 3: Output from second example.

15 percent, and the error in meters was 646.12. Computing the initial location belief is a four step process. These steps are outline below.

1. Calculate a value for distance (*d*) based on:
   d_{actual} = 4307.49
   d_0 = 700 -- pos-d_0, in Table Table *III*
   d_1 = 8000 -- pos-d_1

   $$d = 1 - \frac{d_{actual} - d_0}{d_1 - d_0}$$

   Thus, $d = 0.5058$ for Line Number 1 of Example 2.

2. Calculate *m* based on:
   $d = 0.5058$
   $k = 0.8$
   $v = 1.0$
   $j = 1.0$
   $q = 0.0$

   As explained in Chapter III, the parameters to *m* are combined in three intermediate functions (1), (2), (3).

   Using the data above, the result is $m = 0.76$ for Line Number 1, Example 2.

3. Calculate the straight line distance from the AF to the target. The eventual error will be a percentage of this distance. Then calculate an interval from which a random value may be selected. From Chapter III, Subsection F.1.c, this would be the interval [0, $1 - m$(distance to target)]. This randomization is a critical step in giving the AF a chance, even if a remote one, of hitting its target at almost any range. For Line 1 of this example, $1 - m = 0.24$, and the distance is 4307.49 meters. The maximum error, then, is 0.24 x 4307.49 = 1033.79 meters. However, the actual error chosen at random from the interval [0, 1033.79] was 646.12, representing a percentage error of 15 percent.

4. Determine the believed X and Z coordinates by chosing a direction at random from the interval [0, 359] and then using polar arithmetic to obtain the coordinates of the point that is $r = 646.12$ meters from the target. (Note that the direction in compass degrees must be converted to direction on a standard *x,y* plane.) This output is not shown in the example.

The preceding four steps illustrate the general procedure for computing an initial belief about the location of the target. Thereafter, previous beliefs are taken into consideration. Instead of selecting a direction each time from the interval [0, 359], the the algorithm considers the direction from the target to the previous belief when computing a new belief. The new direction is chosen from a narrower interval, 45 degrees, maintaining consistency from one belief to the next. The distance factor, however, is still determined by applying *1 - m* to the actual distance from the AF to the target.

In Example 2 the target was moving at the same speed as in Example 1; however, the AF had a much harder time destroying its target. In Example 1, the AF was able to destroy its target from a range of 1713.22 meters. But from a comparable range in Example 2, 1683.18 meters on Line 27, the AF had to contend with a six percent error. Since the AF tanks had line-of-sight, they were firing at the target, but they were firing at the wrong location. The AF's missiles could be observed landing near the belief instead of the true target. Obviously, this favors the AF's opponent, who now has much more time to destroy the AF platoon before being destroyed.

Notice in the Error column that some of the entries equal 0. such as Lines 17 and 28. This means that, due to the randomization of *m*, the AF had perfect knowledge at those points in time. Theoretically, the target could have been destroyed at those times. However, even though the AF was firing throughout most of this engagement, it may not have fired at that particular moment due to some obstacle in the terrain, such as a small hill. Or it may have fired, and missed, due to the age of its information, or due to evasive action by the target. Because of network backlogs, the AF sometimes makes decisions based on information that may be too old to be of value. So, even though the belief matches the actual value that came into the program, the actual value itself may be outdated. This implementation problem is discussed in Chapter V.

Line 33 of the output shows the destroyed vehicle type. Even though actual error was one percent, that was sufficiently small to allow a hit.

D. EXAMPLE THREE

Example 3 is very similar to the previous example. The key difference is that knowledge (k) = 0.3, which means that the AF will have less of a chance of scoring a hit in the early stages of the battle. This is evident from Line 2 of the output, shown in Figure 4.

At a range of 4273.60 meters, $m = 0.39$, and the actual percentage error was 34 percent for a distance error of more

```
                KNOWLEDGE TO BELIEF CONVERSION

     Introducing greater location error into the SAF input.

              k = 0.3   v = 1.0   j = 1.0   q = 0.0

 Line
Number  Veh ID   Time       Type    Range     Error    %Error   m

  1.      29   698729.37    53    4343.81   2041.59   0.47   0.38
  2.      29   698737.94    53    4273.60   1453.02   0.34   0.39
  3.      29   698748.94    53    4185.36   1883.41   0.45   0.40
  4.      29   698757.62    53    4114.15   863.97    0.21   0.41
  5.      29   698766.94    53    4037.62   1493.92   0.37   0.43
  6.      29   698774.25    53    3978.40   1233.30   0.31   0.43
  7.      29   698783.94    53    3883.89   660.26    0.17   0.45
  8.      29   698792.56    53    3805.56   1674.45   0.44   0.46
  9.      29   698800.81    53    3723.01   1079.67   0.29   0.47
 10.      29   698812.25    53    3609.88   974.67    0.27   0.49
 11.      29   698821.12    53    3507.24   1052.17   0.30   0.51
 12.      29   698830.75    53    3412.57   1057.90   0.31   0.52
 13.      29   698839.12    53    3316.35   232.14    0.07   0.54
 14.      29   698847.69    53    3230.81   1421.56   0.44   0.55
 15.      29   698856.31    53    3143.71   408.68    0.13   0.57
 16.      29   698865.12    53    3040.42   1003.34   0.33   0.58
 17.      29   698873.94    53    2952.94   413.41    0.14   0.60
 18.      29   698883.37    53    2858.69   829.02    0.29   0.61
 19.      29   698894.50    53    2748.46   659.63    0.24   0.63
 20.      29   698905.62    53    2619.07   209.53    0.08   0.65
 21.      29   698915.62    53    2522.75   807.28    0.32   0.67
 22.      29   698925.81    53    2424.21   169.69    0.07   0.69
 23.      29   698937.12    53    2314.55   624.93    0.27   0.70
 24.      29   698947.81    53    2211.88   575.09    0.26   0.72
 25.      29   698958.81    53    2106.84    0.00     0.00   0.74
 26.      29   698968.81    53    2010.02   482.41    0.24   0.76
 27.      29   698978.31    53    1901.58   133.11    0.07   0.78
 28.      29   698988.75   153    1823.82    0.00     0.00   0.79
```

Figure 4: Output from third AF example.

than 1450 meters. Compare this to Line 2 of Example 2. At a comparable range of 4276.28 meters, *m = 0.76*, and the resulting distance error was 983.54 meters. Of course, comparing random values for two cases is not very informative. However, the 37 percent difference in *m* values creates the opportunity for greater error.

The impact of the difference in *m* values can be shown through simple analysis of the data in Examples 2 and 3. In Lines 1 through 26 of Example 2, the average distance error is 300.82 meters; but in Example 3 the average error for the first 26 lines is 896.35, almost three times as much. This is an indicator of the impact over time of setting knowledge to 0.3 instead of 0.8. In this comparison, lowering knowledge by more than 50 percent resulted in almost three times as much average error. In spite of these odds, the AF destroyed its target from a greater distance in Example 3 than it did in Example 2. In Line 28 of Example 3, the AF exploited an error of 0 by firing at the target when it had perfect location information. This is an example of how randomizing *m* can give the AF a chance of success even when odds do not favor it.

A further comparison of the two examples is provided in Figure 5. For each example, the actual error is plotted as a function of the range to the target. The graph of $k = 0.3$ rises much more steeply than the graph of $k = 0.8$.

E. SPEED AND HEADING ERROR

The next two continuously-valued attributes that merit discussion are speed and heading, both of which are passed to the decision-making portion of the AF program and used in aiming at a moving target. Errors in these attributes are

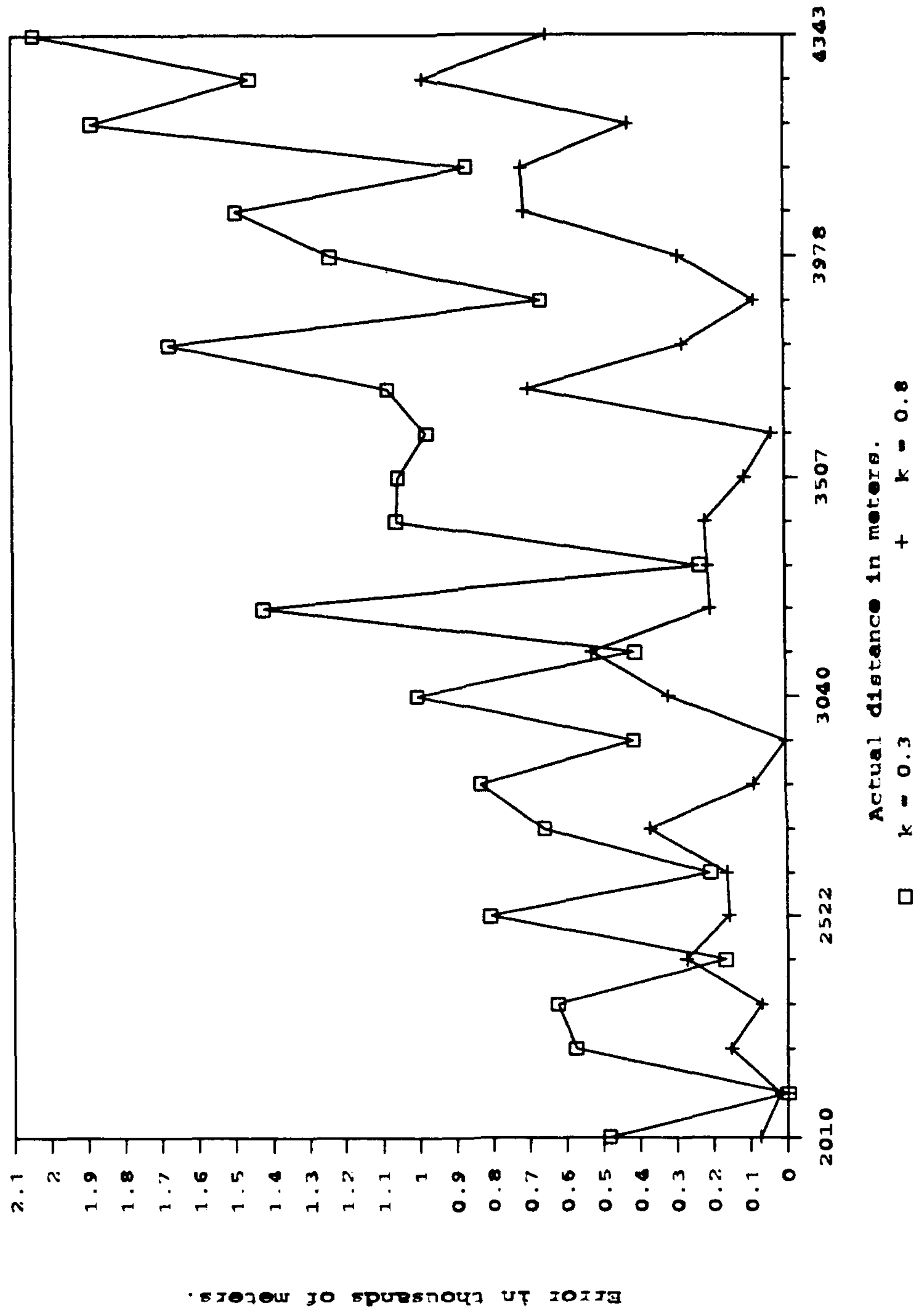


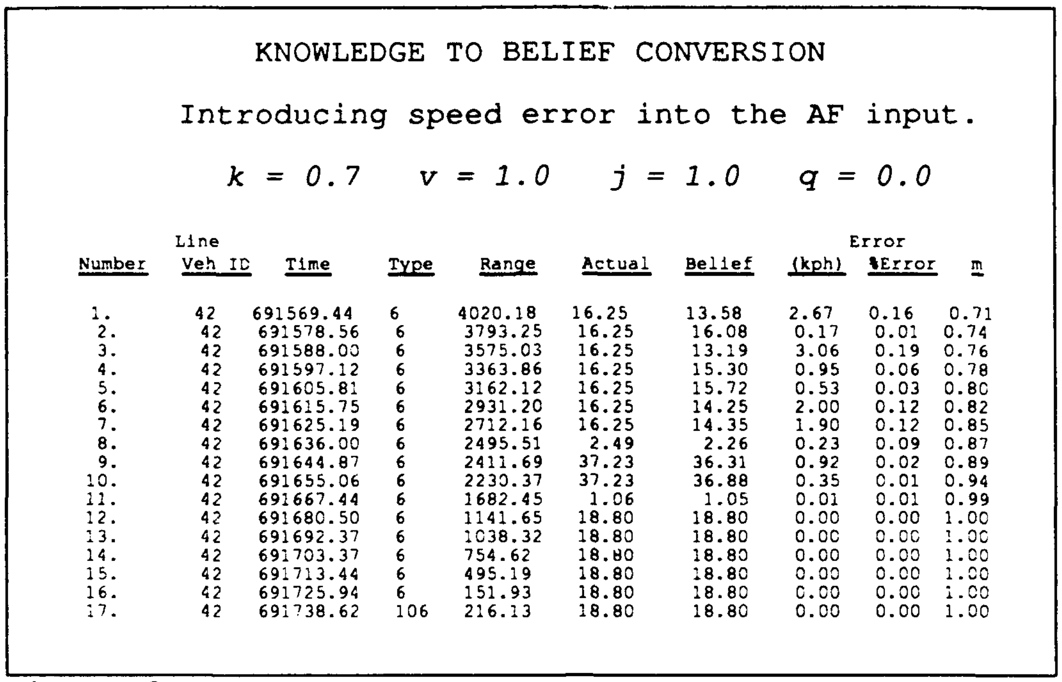
Figure 5: Comparison of two simulations with different values for *k*.

especially significant when the target is moving fast and changing direction frequently. To successfully aim at moving targets, the AF relies heavily on precise, and current, information. Unfortunately, the network delays alluded to earlier become even worse when users of the simulator are making frequent course and speed changes. Thus, the AF has the poorest information during those moments when the information is most critical. To further distort speed and heading information by converting them to beliefs merely compounds the existing error. However, solutions to the network traffic problem are treated as a separate issue. Consequently, the generation of beliefs for speed and heading are discussed here as though the AF would otherwise have perfect information.

1. Speed Error

The first thing to note about Example 4, shown in Figure 6, is the target's overall behavior with regard to speed.

The column labeled Actual shows the target's speed as steady at 16.25 kilometers per hour for several cycles, Lines 1 through 7. This is followed by a sudden drop to 2.49 kph and a sudden increase to 37.23 kph. However, the target is not really accelerating from 2.49 kph to 37.23 kph in a single cycle. That aberration is more than likely due to lost state messages from the simulator. Each change in a



```
                 KNOWLEDGE TO BELIEF CONVERSION

             Introducing speed error into the AF input.

              k = 0.7    v = 1.0    j = 1.0    q = 0.0

          Line                                                        Error
Number    Veh ID    Time        Type    Range      Actual    Belief    (kph)   %Error    m

  1.       42    691569.44      6      4020.18    16.25      13.58      2.67    0.16    0.71
  2.       42    691578.56      6      3793.25    16.25      16.08      0.17    0.01    0.74
  3.       42    691588.00      6      3575.03    16.25      13.19      3.06    0.19    0.76
  4.       42    691597.12      6      3363.86    16.25      15.30      0.95    0.06    0.78
  5.       42    691605.81      6      3162.12    16.25      15.72      0.53    0.03    0.80
  6.       42    691615.75      6      2931.20    16.25      14.25      2.00    0.12    0.82
  7.       42    691625.19      6      2712.16    16.25      14.35      1.90    0.12    0.85
  8.       42    691636.00      6      2495.51     2.49       2.26      0.23    0.09    0.87
  9.       42    691644.87      6      2411.69    37.23      36.31      0.92    0.02    0.89
 10.       42    691655.06      6      2230.37    37.23      36.88      0.35    0.01    0.94
 11.       42    691667.44      6      1682.45     1.06       1.05      0.01    0.01    0.99
 12.       42    691680.50      6      1141.65    18.80      18.80      0.00    0.00    1.00
 13.       42    691692.37      6      1038.32    18.80      18.80      0.00    0.00    1.00
 14.       42    691703.37      6      754.62     18.80      18.80      0.00    0.00    1.00
 15.       42    691713.44      6      495.19     18.80      18.80      0.00    0.00    1.00
 16.       42    691725.94      6      151.93     18.80      18.80      0.00    0.00    1.00
 17.       42    691738.62    106      216.13     18.80      18.80      0.00    0.00    1.00
```

Figure 6: Output from speed error example.

vehicle's speed triggers a message to all stations on the network. Therefore, accelerating rapidly results in overfilled message buffers. From Line 10 to Line 12, the speed drops, then rises, and finally settles at 18.8 kph. Although some speed information has clearly been lost, the emerging pattern throughout the engagement is identifiable--frequent stops and starts and sudden changes in acceleration.

The <u>Beliefs</u> column shows the speed belief calculated for each cycle. As in the previous belief generation processes, the program first calculates a value for *d* using the standard formula, but with different inputs. From Table III these inputs are:

$\text{speed-}d_0 = 1000$
$\text{speed-}d_1 = 8000$

Using Line 1 as an example, $d_{actual} = 4020.18$, and the resulting value for d is 0.57. The remaining parameters are shown in the example. Since $m = 0.71$ for Line 1, the next step is to introduce up to $1 - m$, or a 29 percent, error in the speed belief. Continuing with Line 1 as an example, the speed-error function accomplishes this by computing an interval around the true speed value and then selecting a number from this interval, as explained in Chapter III. The interval is based on $1 - m$. If $1 - m = 0.29$, and the actual speed is 16.25, the interval would be

$$lb = 16.25 - 0.29(16.25) = 11.54$$
$$ub = 16.25 - 0.29(16.25) = 21.0,$$

thus allowing an opportunity for maximum error above or below the true speed, or no error at all if the true speed happens to be selected. In Line 1 of the example, the speed selected was 13.58 kph, which is a 16 percent error.

Since speed error is a percentage of the true speed, and since military combat vehicles do not move at extremely high rates of speed, the belief speeds are generally close to the actual speeds. A quick scan of the column showing error in kph shows that the maximum error in this example was 3.06 kph, on Line 3. Most of the errors were under 1 kilometer per hour. The effect that these errors have on aiming at a moving target depend on the range to the target, the speed of the AF tanks, and the speed of the target in relative terms. That is, a fast-moving target will always be harder to hit than a slow-moving one. Using percentages to model speed-error mirrors this fact because a given percentage of a higher speed is greater than the same percentage of a slower speed.

2. Heading and View Direction Error

The final example in this chapter is used to present a few key points about the direction belief algorithm used by the program. This algorithm is used twice, once to determine a belief about the direction in which the target is traveling, and again to determine a belief about which way the target is facing, or the view direction. The view direction is important because that is the direction in which the target fires its weapon. As with speed information, heading information is used by the AF to aim at a moving target.

Figure 7 depicts the output for Example 5; as usual, the input parameters are displayed with the output. The vehicle type and range columns have been omitted, but that information may be found in the output for Example 2. Both sets of data are from the same simulation.

The heading belief algorithm relies on *m* to determine belief. Initially, the program randomly selects a belief that contains as much as *1 - m* of error. However, subsequent calculations use previous beliefs and the current true value to determine the error. Each subsequent belief is chosen from the interval formed by taking the difference between the last belief and the current true heading. Therefore, if the true heading remains constant, the beliefs will gravitate, randomly, toward the true value. Also, beliefs that are close to the truth will force subsequent beliefs to be even closer.

The most significant feature of this algorithm is its behavior over time. The entries in the Actual column show the target moving for several cycles in a constant direction, 304.73 degrees. After that it moves in a new general direction, with slight changes between 226 and 229 degrees, for the next 21 cycles. Finally, it heads in a general direction of 234 degrees for the last seven cycles.

When the target does not change course, the beliefs tend toward the actual value within a relatively short time. When the target changes course, the beliefs are distorted at first, but then again tend toward the actual value. If a belief happens to match the actual value due to randomization, then the AF will keep that belief until the target changes course again.

KNOWLEDGE TO BELIEF CONVERSION

Introducing heading error into the AF input.

k = 0.8   *v = 1.0*   *j = 1.0*   *q = 0.0*

Line Number	Veh ID	Time	Actual	Belief	Error (degrees)	m
1.	31	697819.81	304.73	299.73	5.00	0.77
2.	31	697826.37	304.73	306.56	1.83	0.78
3.	31	697832.94	304.73	304.08	0.65	0.78
4.	31	697840.56	304.73	304.65	0.08	0.78
5.	31	697847.25	304.73	304.76	0.02	0.79
6.	31	697855.69	226.72	255.27	8.54	0.79
7.	31	697862.75	227.56	227.56	0.00	0.80
8.	31	697869.75	227.56	227.56	0.00	0.80
9.	31	697876.75	227.56	227.56	0.00	0.81
10.	31	697883.69	227.56	227.56	0.00	0.82
11.	31	697890.87	227.56	227.56	0.00	0.82
12.	31	697898.25	227.56	227.56	0.00	0.83
13.	31	697905.25	227.56	227.56	0.00	0.84
14.	31	697912.37	227.56	227.56	0.00	0.84
15.	31	697920.06	227.56	227.56	0.00	0.85
16.	31	697927.81	227.56	227.56	0.00	0.86
17.	31	697935.69	227.56	227.56	0.00	0.87
18.	31	697945.06	227.56	227.56	0.00	0.88
19.	31	697954.25	227.55	227.54	0.00	0.88
20.	31	697963.19	227.40	227.38	0.02	0.89
21.	31	697972.06	228.76	228.71	0.04	0.90
22.	31	697982.94	228.81	228.81	0.00	0.91
23.	31	697995.44	228.95	228.89	0.05	0.92
24.	31	698009.12	228.95	228.93	0.02	0.93
25.	31	698020.06	228.95	228.95	0.00	0.94
26.	31	698034.00	228.95	228.94	0.00	0.95
27.	31	698042.37	234.60	232.64	1.96	0.96
28.	31	698052.00	234.60	235.11	0.50	0.97
29.	31	698060.62	234.60	234.43	0.18	0.97
30.	31	698069.12	234.60	234.60	0.00	0.98
31.	31	698078.50	234.51	234.51	0.00	0.99
32.	31	698088.25	234.51	234.51	0.00	0.99
33.	31	698097.62	234.51	234.51	0.00	1.00

Figure 7: Output from heading error example.

Example 5 provides an illustration of how this works. Line 1 shows an initial belief of 299.73 degrees for heading, which was 5 degrees off the actual heading. At the time, *m = 0.77*. In Lines 2 through 5, the amount of error, in degrees, becomes smaller as the belief heading values approach the actual heading, which remains constant. Then there is a sudden change in heading in Line 6 and a corresponding increase in error. This agrees with the notion that an observer will probably be unsure of the direction of a vehicle immediately after it changes course. However, in Line 7, the belief randomly matched the truth. In this algorithm, when a previous belief equals the current heading value, the belief is retained. Therefore, the next 11 beliefs contain no error. When the target changes course slightly in Line 19, some error is introduced. When the target makes a greater course change in Line 27, even more error is introduced, and then the pattern from Lines 1 through 5 repeats itself.

View direction beliefs are generated the same way as heading beliefs. In the course of most simulations, however, the target maintains a constant view direction, that which is toward the AF platoon. Since the output on view direction is very static a separate example is not presented.

F. SUMMARY

In this chapter, five examples were used to illustrate the generation of location, speed, heading, and, indirectly, view direction beliefs. The calculation of location beliefs was discussed in detail, with examples of how *d* and *m* are computed. Actual output from a simulation was used to show the workings of the algorithms. In the next chapter, the implementation of this program will be discussed.

V. IMPLEMENTATION

The AF application described in this thesis was implemented as a single program with two distinct phases: the observation and belief generation phase, and the decision-making and execution phase. Program operation is sequential, with each phase executing in turn. The phases were implemented as separate rule-based programs using the C Language Integrated Production System (CLIPS). The two CLIPS programs were embedded in a main program written in C. In this chapter, the organization of the overall program is described in more detail.

A. ORGANIZATION AND SEQUENCE OF EVENTS

NPSNET can be operated on one or more graphics workstations that are connected to a network. The AF program, when it is used, must be on the same network, but loaded on a workstation that is not running NPSNET.

Before starting the AF, a network interface program must be started. This program provides the low-level functions needed by the AF to operate as an independent station on the network.

Upon starting the AF, there is a brief network initialization period and then the user is given the chance to turn belief generation on or off. If it is turned off, then the decision-making phase of the program will be passed incoming messages exactly as they are received from NPSNET. If belief generation is turned on, each message will be processed into beliefs.

The next step in the program sequence is to start the CLIPS environment. This permits CLIPS function calls, such as "load" and "run," directly from a main program**[Ref. 14]**. Following that is a second initialization period during which the user is prompted for the type of mission the AF is to perform and the starting location of the AF.

There are presently three basic missions the AF can perform: reconnaissance, attack, or a user-defined mission. The reconnaissance mission is applicable to most simulations because it sets the AF platoons on a predetermined course which can be intercepted by the user to stage an engagement. The starting location consists of X and Z coordinates. They can be anywhere in the NPSNET world space.

If desired, the user can also change the settings of the parameters in the belief generation program. Chapter IV contains a discussion of how the settings impact on program operation. These settings are defined as *defglobals* in the file *afbelief.clp*, which contains the CLIPS belief generation program.

After all initialization, the main decision loop of the program begins. The sequence of events within the loop is as follows:

1. Update the positions of all AF vehicles and target vehicles, based on their last reported location and speed.

2. If a new message is present, get the message off the network; store the data in a temporary data structure.

3. Load the belief generation program into the CLIPS environment, followed by the most current target data and AF data. The most current target data will normally be the contents of the latest state message. But, if no message has arrived, the most current information is that contained in the updated target data structures.

4. Start the CLIPS program. It will run until no rules remain on its agenda. While running, the CLIPS belief generation program will load external data structures with the beliefs that have been calculated. Since these structures are defined outside of CLIPS, the data is retained after the CLIPS program has run its course.

5. Start the decision-making phase. This is the second CLIPS program, loaded in the same manner as the first. Input is transferred from the data structures that were loaded by the belief generation program. The output from this phase is sent to NPSNET as an update message.

6. Continue looping until control-c is pressed.

An important characteristic of this organization is that the CLIPS files are reloaded and run during each decision loop. Although this impacts on the speed of execution, it permits modification of either of the CLIPS programs during execution. For example, if the value for *knowledge* in the belief generation program is changed, and the file is saved, the new value of that parameter will be loaded during the next decision loop. Although this feature is useful for program test and evaluation, it is not optimal to have the main program read files from a disk during each decision loop. CLIPS provides a means of compiling the constructs into the main program so that it is entirely self-contained.

The present version of the AF program includes a special feature for tracking the behavior of the belief generation program. When the program loads the proper data structure with belief information, a copy of the data is sent to NPSNET so that the belief can be displayed on the screen. To avoid confusion, the belief is represented by a unique object. This provides a visual picture of, for example, where the AF *believes* a target is located.

B. SUMMARY

Although the implementation of the AF is somewhat simplistic, it lays the foundation for future autonomous agent programming at the Naval Postgraduate School. Successive versions will become more sophisticated as researchers attempt to model more complex AF behavior (see Chapter VI). To handle this greater complexity, other implementation approaches might be tried, such as coding the entire program in a procedural or object-oriented language instead of a rule-based language.

From a hardware point of view, future refinements may involve parallel processing of the AF phases, a better approach to linking the AF with NPSNET (i.e., networking issues), and experimenting with other platforms.

VI. CONCLUSIONS

While conclusions about implementation details were discussed in Chapter V, this chapter focuses on the merits of the model itself. First there is a review of the original goals and assumptions and then an assessment of this approach to modeling combat observation systems.

A. A REVIEW OF THE INITIAL ASSUMPTIONS AND STRATEGY

In Chapter I it was pointed out that this project began with several key assumptions. The most important of these was the assumption that the AF, when given perfect information, would always hit its target. Therefore, the goal was to allow the AF to operate with beliefs instead of the true world state.

The basic strategy was to divide the AF program into two broad functions that model real world combat subsystems: battlefield observation and tactical decision making. The observation function is concerned with how the AF gets its information, and the decision making function chooses courses of action based on that information. Since the input to the AF program was to be NPSNET world state messages, the observation function would have the task of converting those state messages into information that is comparable to what the AF would have realistically been able to obtain.

When modeling the combat observation subsystem, the goal was to simulate the results of the process, not the process itself. This approach was evident in the design of the belief generation algorithms. In general, they sought to simulate the results or consequences of human error without attempting to simulate human thinking or perception.

B. OVERALL STRENGTHS AND WEAKNESSES

Data provided in Chapter IV shows how rapidly the AF can destroy a target when it has perfect information. Testing of the AF program thus far shows that this is true consistently. Therefore, assumptions about the AF's performance with perfect information appear to have been valid, at least when evaluating the AF's performance on NPSNET. This assumption may not apply to all autonomous force applications.

Regarding the choice of strategy, it will be necessary to test the AF program extensively before the strategy can be fully evaluated. At this point in the development, however, the concept of dividing the AF into an observation function and a decision making function appears to produce the desired results. When beliefs are generated, the effectiveness of the AF is measurably degraded. Since this was the original goal, and it was met, then it is tempting to argue that this strategy works.

However, a possible weakness of this model, at least in its current form, is that it may not be suited for modeling a more sophisticated observation system. In a real battlefield environment, there may be a great deal of dialogue between decision makers and observers. When the decision maker has a question, the observer can be queried. Also, the observer's reports can be extremely detailed and subjective, and may contain interpretations of what is seen on the battlefield, not just descriptions. The current approach, with its emphasis on results instead of processes, would have to be redesigned to model more of the human thought processes that go into the acts of observing things and reporting observations to others.

On the other hand, a model with as many features as just described might no longer be appropriate for a real time application. Furthermore, even in the real world, small combat units such as tank platoons do not rely on extremely sophisticated information gathering methods.

Given these arguments, it is probably fair to say that this strategy is right for a relatively simple, small-unit autonomous force application, but may not be a good general approach to modeling the battlefield observation system as it exists in the real world.

C. RECOMMENDATIONS FOR FURTHER DEVELOPMENT

The AF program will continue to be a valuable research vehicle for others wishing to investigate autonomous agent issues. Among the many enhancements that warrant additional research are:

1. Adding line-of-sight computations to the observer program so it can prevent the AF platoons from knowing about objects that are not visible.
2. Adding a vehicle identification feature to the decision making program so the AF can be selective about its target choices.
3. Modeling more battlefield conditions, such as time of day, weather, noise, and the duration of a battle.
4. Refining the vehicle type belief algorithms so that they consider such things as what the vehicle is doing when it is first spotted, the type of terrain the vehicle is driving on (because trucks usually stay on roads), and the view of the vehicle from the perspective of the observer.
5. Experimenting with different reasoning techniques in the discretely valued belief generation procedures.

D. SUMMARY

In this chapter we presented the following conclusion: the initial assumptions are valid and the approach is probably appropriate for a system such as NPSNET. However, this method may be oversimplified as a general purpose strategy for autonomous agent programming.

LIST OF REFERENCES

1. *Webster's New Twentieth Century Dictionary*, Second Edition, Simon and Schuster, 1983.

2. Zyda, M.J., and D.R. Pratt, "NPSNET: A 3D Visual Simulator for Virtual Exploration and Experimentation," *Digest of Technical Papers XXII*, SID International Symposium, 8 May 1991.

3. Brooks, R.A., B.G. Buchanan, D.B. Lenat, D.M. McKeown, J.D. Fletcher, "Panel Review of the Semi-Automated Forces," IDA Document D-661, Institute for Defense Analysis, September 1989.

4. Culpepper, M., *Tactical Decision Making in Intelligent Agents: Developing Autonomous Forces in NPSNET*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March, 1992.

5. Davis, E., *Representations of Commonsense Knowledge*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1986.

6. Hintikka, J., *Knowledge and Belief*, Cornell University Press, 1962.

7. Konolidge, K., *A Deduction Model of Belief*, Pitman, London, 1986.

8. Neapolitan, R.E., *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & Sons, Inc., New York, 1990.

9. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufman Publishers, San Mateo, CA, 1988.

10. Jackson, P., *Introduction to Expert Systems*, Addison-Wesley Publishing Company, 1990.

11. Buchanan, B.G., and E.H. Shortliffe (ed.), *Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, 1984.

12. Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.

13. Buchanan, B.G., and E.H. Shortliffe (ed.), *Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, 1984.

14. *CLIPS Reference Manual, Volume II: Advanced Programming Guide*, Version 5.0, Software Technology Branch, Lyndon B. Johnson Space Center, 1991.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	U.S. Army Artificial Intelligence Center Pentagon Rm 1D659 ATTN: CSDS-AI (Proponency) Washington, DC 20310-0200	1
4.	Mr. John E. Laird Artificial Intelligence Laboratory University of Michigan 1101 Beal Ave. Ann Arbor, MI 48109-2110	1
5.	Mr. David Pratt Naval Postgraduate School Code CS/Pr, Computer Science Dept. Monterey, CA 93943-5100	1
6.	Institute for Defense Analysis Simulation Laboratory 1801 N. Beauregard St. Alexandria, VA 22311	1
7.	U.S. Army CACDA ATTN: ATZL-CAC-FBL Fort Leavenworth, KS 66027	1
8.	Professor Hemant Bhargava Naval Postgraduate School Code AS/BH, Administrative Science Dept. Monterey, CA 93943-5000	1
9.	U.S. Army CAA ATTN: CSCA-RSR 8120 Woodmont Ave. Bethesda, MD 20814-2797	1

10. CPT Mike Culpepper 1
Route 1, Box 32
Malvern, AR 72104

11. CPT William C. Branley 1
P.O. Box 3658
Merrifield, VA 22116-3658

12. Professor Tung Bui 1
Naval Postgraduate School
Code AS/Bd, Administrative Science Dept.
Monterey, CA 93943-5000