Comput. Math. Applic. Vol. 13, No. 1-3, pp. 261-274, 1987
Printed in Great Britain. All rights reserved

0097-4943/87 $3.00 + 0.00

ARTIFICIAL INTELLIGENCE IN AIR COMBAT GAMES

E. Y. RODIN, Y. LIROV and S. MITTNIK
Department of Systems Science and Mathematics, Washington University in St Louis, St Louis, MO 63130, U.S.A.

B. G. MCELHANEY and L. WILBUR
North American Aircraft Operations, Rockwell International, El Segundo, CA 90245, U.S.A.

Abstract—A general framework for the utilization of large numbers of optimal pursuit-evasion algorithms, as applied to air combat, is described. The framework is based upon and is driven by artificial intelligence concepts. The method employed involves the valuation of alternative tactical strategies and maneuvers through a goal system and pilot-derived expert data bases. The system is designed to display the most promising strategies to the pilot for a final decision.

Two aspects of the concept above are described here: the general framework and a specific implementation for a synthetic method of flight and fire control system optimization. Details of the implementation, based on off-the-shelf hardware and a standard programming lanuage, are also given.

Potential utilization of these concepts includes other areas as well: submarine warfare and satellite based weapon systems are two possible additional applications. Nonmilitary applications are air traffic control and optimal scheduling.

1. GENERAL OVERVIEW

1.1. Introduction

Over the past several years, analyses and simulations have disclosed several important problem areas in air combat which must be solved to exploit fully advanced airframe, sensor and weapon technologies planned for future fighter aircraft. The advent of advanced medium range air-to-air missiles and projections of supersonic cruise and maneuverability for future fighters have substantially increased the tactical fire control solution space and complicated problems of air combat. The medium range complexities are caused by such things as lack of overall situational awareness on the part of the pilot; the guidance and control characteristics of medium range air-to-air missiles; the performance of advanced multi-sensor suites; short decision/action time lines at supersonic speeds, and the fact that the pilot must rely entirely on sensor supplied information to assess his opponents behavior and reach a fire control decision. Also, because the medium range combat can become very rapidly a close-in-combat, there will be many situations in which short range missiles and guns will become the principal weapons in the engagement. In the close-in environment, the principal complexities are the extreme compression of decision/action time lines, and the workload of controlling the aircraft and making complex tactical decisions under high mental and physical stress conditions. The basic air control problems are further complicated by the presence of the ground-to-air fighting facilities as well as the topographical constraints.

Air combat may be considered as a game problem in which opponents mutually endeavor to maximize their opportunities to destroy an enemy and to minimize their own risk of loss. The conduct of the game is based on mutual information about each others total weapon system performance and capabilities, actual position in space, current velocity vector, ownship and opponents energy state, etc. In current classical air combat at short range, pilots have been using their eyes as sensors and their brains to integrate the visual and sensor supplied information necessary to play the game. The basic processes used by the pilot have been codified in a concept called the OODA loop—Observe, Orient, Decide and Act—in a cyclical manner. The OODA loop process and its relationship to aircraft functions is shown in Fig. 1. While the OODA loop will undoubtedly continue to play an important role in air combat pilotage, the limitations are obviously those of human ability in the aforementioned supersonic combat environments. Humans are characterized by a limited processing rate [1]—two events occurring closer together than about one tenth of a second will generally be perceived as a single event.

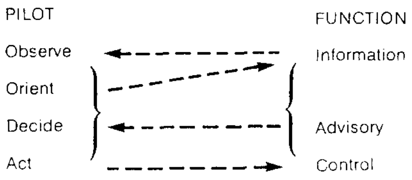


Fig. 1. The OODA loop process.

The same time scale roughly holds for elementary cognitive and motor processes with a range of 25–200 ms per operation. An activity that requires integrated perception, decision and motor action requires on the order of one-half second and requires processing of something like 10–40 bits per second of information. A more complicated activity in which problem solving is involved, such as air combat against multiple targets, is likely to require up to five seconds per problem step. A final limitation on processing rate is that the human, with few minor exceptions, is a serial information processor. Hence, attention to two or more activities requires rapid switching between tasks, creating opportunities for error or misjudgement.

In future air combat and in medium range combat in particular, target information will be gathered principally by electronic sensors. Then the crucial questions will be:

how to transmit information to the pilot?
how to use computers to assist the pilot in playing the combat game?

There are two extreme concepts for potential solution. First, the sensor data could be used to generate a suitable situation display upon which the pilot could act as he currently does in classical air combat, using his brain on an artificially generated and displayed situation. Second, sensor data could be used as input to a tactical computer which generates and displays commands about how the aircraft should be maneuvered and how (and when) weapons should be deployed. Presumably, neither of the alternatives is entirely feasible today. Because of the overall complexity of the air combat problem and human information processing limitations the pilot may not be able to play the game effectively, even if a perfect way of displaying tactical situations is found. The prerequisite for a computer to play the game is the mathematical solution of the air combat game problem in real time. However, air combat pursuit and evasion games have shortcomings and present computer technology may not be able to handle the speed and memory requirements needed for the real-time computation of solutions. Most of the current available pursuit–evasion algorithms suffer from overly simplistic formulation and too many abstractions in order to permit appropriate mathematical solutions. For example, it is usually assumed that at every moment during the fight pursuer or evader roles can be assigned to every aircraft and that, according to the role assigned, a suitable (i.e. optimal in some cases) trajectory is computed. In reality however, these roles are not necessarily fixed during a mission and a fighting plane may switch several times from a pursuing to an evasive maneuver and vice versa. Hence, the question of either pursuing or evading may be irrelevant. It is advantageous to view a multiplayer positional game as a multi-stage game, where every player tries to select an optimal sequence of actions corresponding to the remaining stages of the game. In this formulation the current state of differential game theory appears to be rich enough to be applied during a particular time segment. Solving multiple differential games and lumping the solutions together becomes, however, a difficult computational task. Here an Artificial Intelligence approach to designing an operational on-board system, combining the extreme concepts aforementioned, is proposed.

1.2. Expert system in aerial combat—an outline

The aim of this investigation is to design a Tactical Decision Aid Expert System (TDAES) supporting pilots in tactical decision making processes [2].

A simplified block diagram of the TDAES reported upon here is shown in Fig. 2. The figure illustrates the TDAES synthesis of the pilot centered, non-automated OODA loop process as described earlier.

The Expert System will generate an initial optimal flight and action plan (initial mission). The optimal plan will be reevaluated and possibly changed every time when some unforeseen event takes

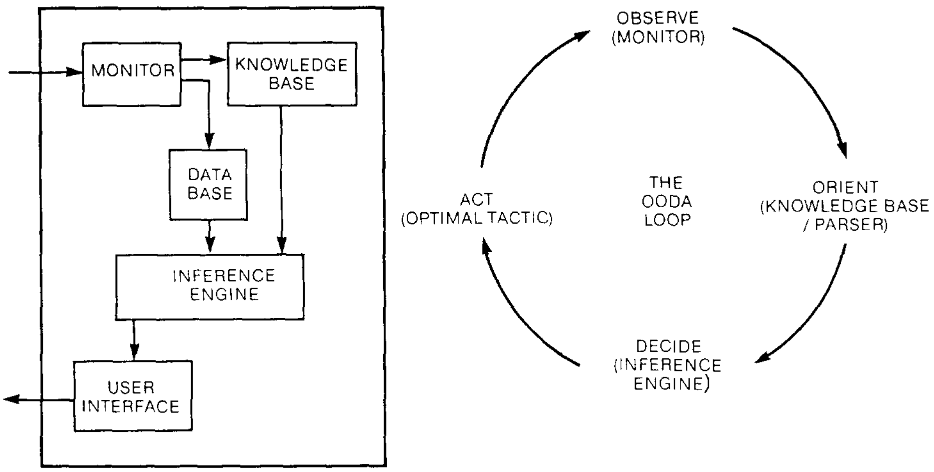


Fig. 2. TDAES block diagram.

place. The system will employ a basic set of pursuit–evasion algorithms for optimal mission generation. A fuzzy goal function [3] will be constructed for the evaluation of the plans permitting the player to take "reasonably offensive and not self destructive" roles. The main components of an Expert System performing the OODA loop process are [4]:

—Data Base
—Knowledge Base
—Inference Engine
—User Interface
—Monitor.

The *Database* is used for storage and retrieval of the necessary information about the participating fighter capabilities, the topographical environment, as well as the assumed future actions and positions of the players. Specifically, it contains a pre-mission data file which includes:

—The importance values of the participating players, both at the strategic and at the tactical levels.
—Information on own capabilities, including fuel, missiles, ranges, speed, turn radii, etc.
—Geographical information.
—Available information on enemy capabilities.

The *Knowledge Base* is used for storage and retrieval of pursuit–evasion algorithms and additional data allowing the Inference Engine to define the optimal sequences of actions (missions) and flight trajectories, as well as an Expert Data file based on past pilot experiences.

The *User Interface* is utilized for display of the answers to the following types of questions:

—What mission?
—What action?
—What is the next control?
—Why such an action?

In a more refined mode, it would also display information of the following sort:

—What is the stability of the action proposed?
—What alternative actions might be available?

In simulation the *User Interface* has the additional tasks of:

—Graphical display of the players in real time,
—Accepting input from the peripheral devices (e.g. a mouse).

The *Monitor* function is to receive an input data stream from various aircraft sensors such as a fire control radar trackfile or a multi-sensor corporate fire control trackfile developed from fused

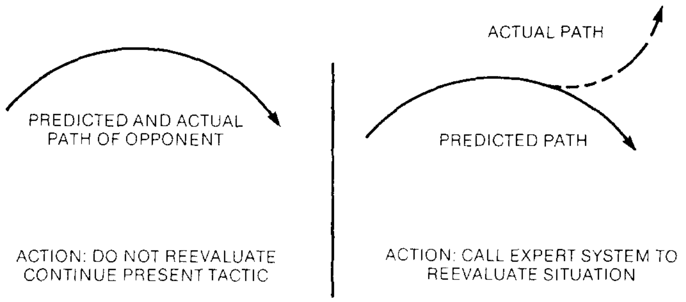

PREDICTED AND ACTUAL PATH OF OPPONENT
ACTION: DO NOT REEVALUATE CONTINUE PRESENT TACTIC
ACTUAL PATH
PREDICTED PATH
ACTION: CALL EXPERT SYSTEM TO REEVALUATE SITUATION

Fig. 3. Example monitor operation.

sensor data [5] and to compare the current states of all targets in track with the states of those same targets predicted by the TDAES. If the states match within predicted limits, the maneuver being employed at the time is continued. However, when there is divergence, i.e. when the state of the tracked target deviates from the prediction, the monitor calls the TDAES to reevaluate the tactical situation (Fig. 3).

Finally, the purpose of the *Inference Engine* is to generate an ordered list of sequences of actions.

2. IMPLEMENTATION

2.1. Introduction

In the remainder of this paper we are reporting on the specific implementation of the ideas mentioned in the previous section. It is necessary to look at this implementation from two points of view: its actual current status is one of them, and its expansion potential is the other.

The system was developed by using minimal off-the-shelf hardware—an IBM PC AT, with 0.5 Mbyte memory and with an auxilliary mouse. The language used was Turbo Prolog 1.0. The length of our current code is about 3500 lines.

Now it is quite clear that in any active working implementation of our ideas the appropriate hardware would be a much more sophisticated, special purpose, multiprocessing one; the language would almost certainly not be Prolog and the inputs would be received in ways quite different from a mouse. Yet, it seems to us that, as a benchmark, our system is almost ideal; for we can accomplish almost any of the eventual tasks on it, with the only penalty being the non-real time aspect of its operation. For example, our program can accommodate an arbitrary number of players, with any number of possible actions, and with variable weights for those actions; we can also accommodate almost arbitrary topography and a large variety of weapons systems. An additional feature of the system is its self-tuning capability; a property which, however, has not been utilized yet.

Thus, what we are presenting in the following sections is both the idea and the implementation of a *framework* for the utilization of pursuit–evasion algorithms. This framework, when equipped with the appropriate algorithms, is capable of accepting both corporate and sensor information, evaluate tactical situations based on the above and on its *Knowledge Base*, and propose an optimal action sequence to the pilot. The system also has a statistical debriefing capability after every action and/or simulation.

We note, however, that the version here presented is limited to constant velocities and to two dimensions. On the other hand, the Expert System built into our model is quite efficient, because it was specifically designed for this purpose alone, and it operates in helping each player (whatever its color) evaluate its tactical position. The philosophy behind it derives from optimal utility theory.

2.2. Knowledge Base

The task of the Expert System is to determine the most favorable missions. The search space consists of the missions μ_i which are constructed in the following way:

(1) the set of legal relations is attached to every participating player Π_j,
(2) the set of algorithms to perform every relation is chosen,
(3) using the sets in (1) and (2) the sequences of actions A_{ij} are lumped to generate a mission μ_i.

2.2.1. Definition of mission. Let J be the index set of all players in the game. A mission μ_i is defined as an ordered triple

$$\mu_i = (\Pi_i, \tau_i, L_i)$$

where

Π_i is a given player i, $i \in J$

τ_i denotes time instance i

and

$L_i = (A_{i1}, A_{i2}, \ldots, A_{in})$ is an ordered list of actions of Π_i to all participating players, and n is the cardinal number of J.

An action A_{ij} is an ordered triple $A_{ij} = (g_{ij}, \Pi_j, t_{ij})$,

$g_{ij} \in R_{ij}, j \epsilon J$,

where

R_{ij} is the set of all feasible relations between player Π_i and player Π_j,

and

t_{ij} is the time when the action A_{ij} starts.

Note: $t_{i1} = \tau_i$.

2.2.2. Definition of relations. Two kinds of players are allowed in our system, the dynamic players (e.g. planes) and the static ones (e.g. SAM or obstacle, etc.). The following are examples of possible feasible relations sets:

$R_{ij} = \{$ignore, reach-and-fire, avoid$\}$
when the planes Π_i and Π_j belong to different parties, or
$R_{ij} = \{$ignore-and-fire$\}$
when Π_i is a SAM of one party and Π_j is a plane of another.

The relation "ignore" is a no-action relation. The relation "ignore-and-fire" is the action which a SAM takes against the opponent party's plane: it ignores it as long as it is too far away, and then fires, whenever there is a chance to hit the target. The relation "reach" means literally reaching a point in the space, while "reach-and-fire" is defined as "reach" when the target point is too far and then it is changed to "fire". "Avoid" is always used in conjunction with the next "reach" or "reach-and-fire". The "evade" term used in the differential games field means usually "maximize the time before getting captured" or when possible, "take any trajectory leading to a secure state". We, however, prefer an "avoid" term which we define as a local action taken just to avoid a static or a dynamic object on the way to "reach" the next target.

2.2.3. Knowledge representation. After constructing the missions, the Inference Engine evaluates them using a goal system as described below. However the mission generation task of the Inference Engine may become computationally untractable. Without giving a full computability analysis, let us just consider the following example. Given five participating players, a library of 25 pursuit–evasion algorithms, and three legal relations that a player could assume, there may be $4! \cdot 3 \cdot 25 = 1800$ missions generated.

The evaluation of every mission in such a search space may easily exceed reasonable time limits. Heuristics are applied in order to generate only "reasonable missions", so that the search space becomes greatly reduced. The geometry of the players' current pose, the (precomputed) chances of survival in any encounter of given types of aircraft and ammunition, and the preference specifications given in the goal system determine to what extent a mission is reasonable or not. The production rules are used for representing this type of knowledge in the knowledge base. This representation has the following desirable features:

—(1) *modularity*: every rule represents an independent piece of expertise,

—(2) *incrementability*: the total of the rules can be increased independently,
—(3) *transparency*: the ability to explain its decisions and solutions.

The *Knowledge Base* consists of four sets of rules:

—(1) relation rules,
—(2) algorithm rules,
—(3) sequencing rules,
—(4) selection rules.

We shall now give some examples of these:

```
Relation-rule 1
  if  can intercept the opponent
      and
      can be intercepted by opponent
      and
      chances of survival are high
      and
      chances of destroying the opponent are high
      and
      weight of the opponent is high
then
      relation: = reach-and-fire

Relation-rule 2
  if  cannot be intercepted by opponent
      and
      cannot intercept the opponent
      and
      weight of the opponent is low
then
      relation: = ignore

Algorithm-rule 1
  if  action 1 is "avoid a plane"
      and
      action 2 is a "reach and fire a static target"
then
      algorithm: = perpendicular bisector algorithm
```

Note: *Differential Games* by R. Isaacs, p. 11 [6]

```
Sequencing-rule 1
  if  action 1 is a "reach and fire the plane"
      and
      action 2 is a "reach and fire the ground target"
      and
      the ground target is on the way to the plane
then
      order is (action 2, action 1).

Selection-rule 1
  if  the expected utility of the mission is relatively low
then
      discard it.
```

2.3. *Inference Engine*

The *Inference Engine* performs three tasks:

—(1) generate reasonable missions,

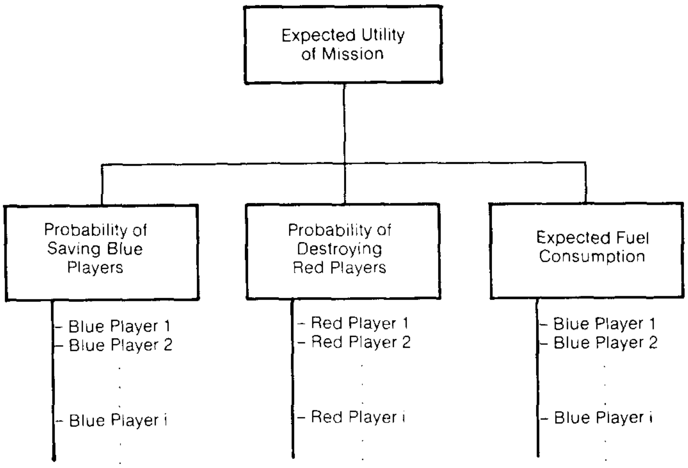


Fig. 4. Illustrative goal system for party "blue".

—(2) evaluate and rank the missions,
—(3) perform sensitivity analysis.

2.3.1. Mission generation. The reasonable missions are generated by actively using the knowledge contained in the *Knowledge Base*. It is basically an exhaustive search following the rules specified in the *Knowledge Base*.

2.3.2. Mission evaluation. To determine the most favorable missions, the *Inference Engine* evaluates every mission generated as described above. The evaluation of a mission yields a scalar value reflecting the expected utility of that particular mission. The ranking of the missions is then performed according to their expected utilities.

Following the approach in Mittnik [7], the expected utility of a mission is derived by using an operational goal system, capturing the objectives and preferences of a party. The goal elements are structured in a hierarchical fashion and attached with weights reflecting their relative importance. A goal system for air combat may contain the following elements (the weights are given in parentheses):

Overall goal	G_0:	maximize the expected utility of the actions taken by the party.
Subgoals Level 1:	$G_{1,1}$:	maximize probability of saving own players (W_S)
	$G_{1,2}$:	maximize probability of destroying enemy players (W_D)
	$G_{1,3}$:	minimize expected fuel consumption (W_F).

The goal elements in the second level of the hierarchy are determined by the participating players. Figure 4 illustrates a possible goal system of party "blue".

A major task of the Expert System is to specify a sequence of actions (mission) for each of the party's players such that the overall objective is maximized. In order to do so, the Inference Engine generates missions and evaluates their respective contribution to each subgoal at the bottom level of the goal system. Let v_i denote the tactical value of a player i, $i \in J$. (The different parties might use different sets of the tactical values.) Let $J = J_R \cup J_B$ where J_R is the set of the red players and J_B is the set of the blue players. Adopting the goal structure in Fig. 4, the overall objective function is defined by

$$EU(\mu_i) = w_S \sum_{j \in J_B} v_j PS_j + w_D \sum_{j \in J_R} v_j PK_j + w_F \sum_{j \in J_B} F_j, \tag{1}$$

where

PS_j probability of survival of the player Π_j during the mission,
PK_j probability of destroying the player Π_j during the mission,
F_j total fuel consumption of the player Π_j.

The weights v_i and W_j are specified in the beginning of the game, while the probabilities PS_j and PK_j and the fuel comsumptions F_j must be recomputed during the game. Let $M = \{\mu_1 \dots \mu_n\}$ be the set of all missions of all players at the given moment. Let T denote the increasingly ordered set of $\{t_{ij}\} = \{t'_e\}_{e=1}^{N}$ of the missions of all the players. Then the stages in the game are defined by the intervals $\tau_e^* = (t'_e, t'_{e+1})$ for all e.

The probability PK_{je} of destroying player j during stage τ_e^* is given by $PK_{je} = 1 - PS_{je}$ where PS_{je} is the probability of survival of player j during the stage τ_e^*. For example, the individual stage probabilities PS_{je}, are aggregated to the total probability PS_j, using

$$PS_j = \prod_{e=1}^{N} PS_{je}. \tag{2}$$

The probability PS_{je} of survival of player j during the stage τ_j^* is computed on the basis of the trajectories of all the players at that stage, their firing envelopes and actions.

2.3.3. Sensitivity analysis. The sensitivity analysis provides the confidence intervals of the expected utilities of the reasonable missions. The robustness of the ranking of the missions can be examined by comparing the worst, the best and an intermediate scenario.

The main decision factor in the favourable mission is traced as well as the reasons for a mission being discarded.

2.4. Current implementation

In our current implementation we allow five types of players for each of the two opposing parties:

—(1) plane
—(2) ground target
—(3) obstacle
—(4) SAM battery
—(5) home.

In addition, the presence of neutral players is also permitted.

The firing envelopes of the offensive players are simulated by lists of destroy probability radii. The planes have two lists:

(1) air-to-air
(2) air-to-ground.

The SAMs have one list: ground to air (Fig. 5). Hence, long, medium, and short range-type missiles can be easily simulated using this approach. The dynamics of the planes in this simulation

TYPICAL PLAYER

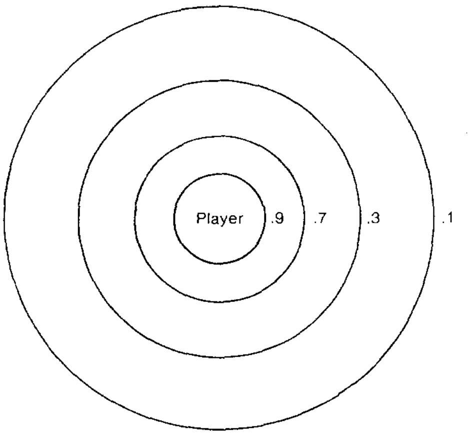


Fig. 5. Probability circles around player.

behave according to the following differential equations:

$$\begin{aligned}\dot{x} &= \omega \sin \theta \\ \dot{y} &= \omega \cos \theta \\ \dot{\theta} &= \frac{\omega u}{R} \\ |u| &< 1\end{aligned} \tag{3}$$

where (x, y, θ) is the pose (i.e. the location and direction) of player. The angle θ is measured from the positive Y axis in a clockwise direction. The player's maneuverability is measured by its minimum turning radius R. The curvature of the player's trajectory is determined by the control u. ω is the speed of the player which here is assumed to be constant.

We adopted the following approach for this game simulation as suggested in Lirov [8]:

(1) Define events as significant changes in the geometry of the game state, e.g. an event took place when the distance δ between the previous and the new locations of any participant is, say $\geqslant 1$.

(2) Create a list of events ordered in time. The time for every player is simulated, i.e. it is computed from the previous time and the speed of the participant. In our example, from (3) it follows:

$$\delta^2 = (x_n - x_{n-1})^2 + (y_n - y_{n-1})^2 = (\Delta t\, \omega)^2 \tag{4}$$

therefore,

$$t_n = t_{i-1} + \delta / \omega \tag{5}$$

for players i, i.e. for different velocities we obtain different time increments.

The simulation program will perform the following loop:

```
repeat
    select first event
    perform
    generate next event
until the game is over.
```

In the next several figures we shall illustrate the principal features of our system. Thus:

The *data flow* is depicted in Fig. 6.

The hierarchy of the *software modules* is illustrated in Fig. 7.

Note that, because of the purposefully fuzzy nature of our formulation, an engagement is almost always going to have a probabilistic outcome; and that, in particular, draws are possible.

The data structure for the *event queue* is presented in Fig. 8.

The data structure for the *players* is given in Fig. 9.

The data structure for *best tactics* is contained in Fig. 10.

The *main menu system*, schematically described in Fig. 11, is further illustrated in the sequence of Figs 12–15.

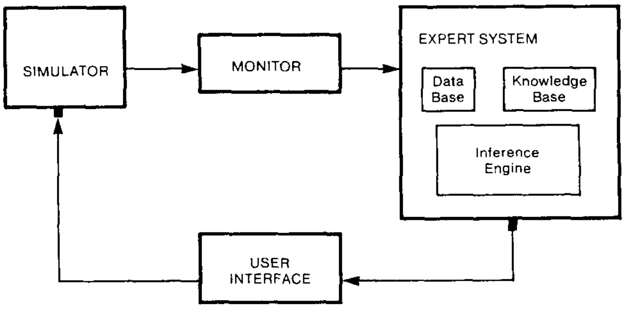


Fig. 6. The data flow.

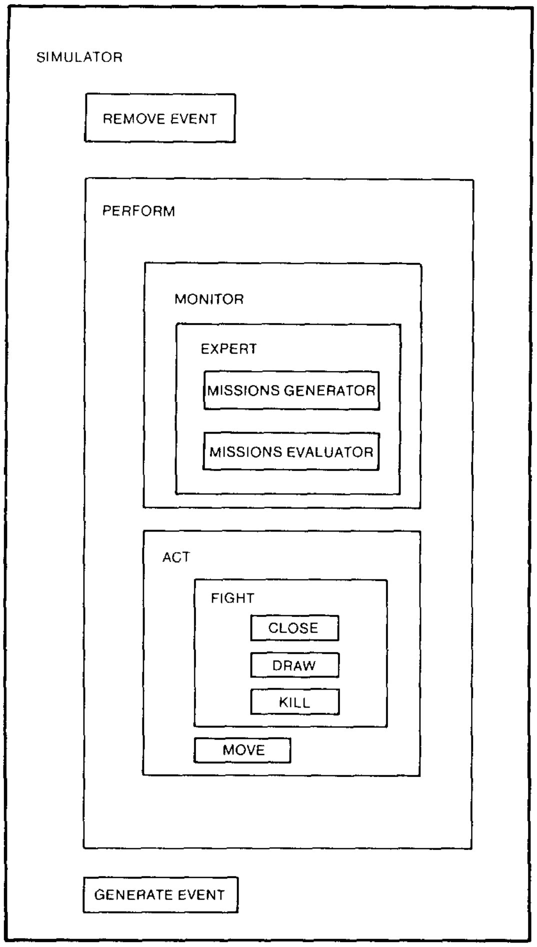


Fig. 7. The hierarchy of software modules.

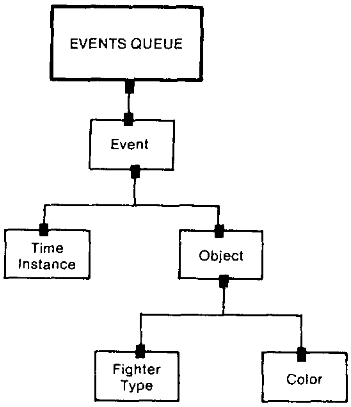


Fig. 8. Event queue data structure.

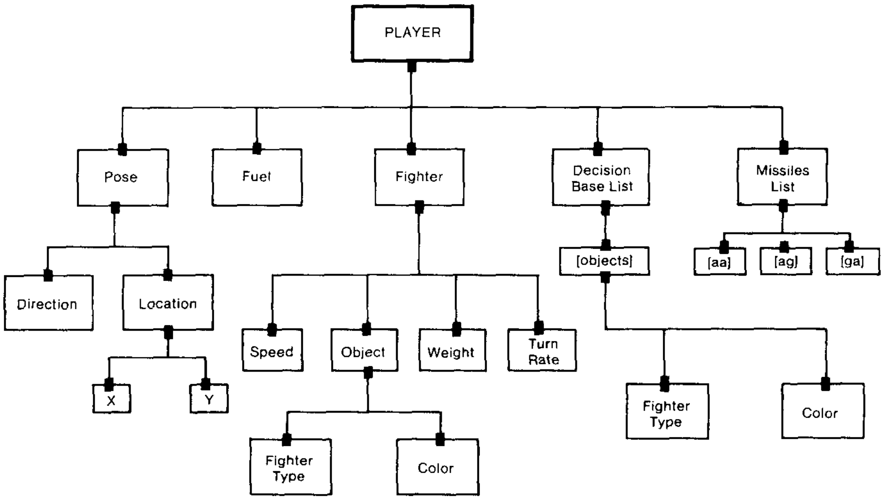


Fig. 9. Players data structure.

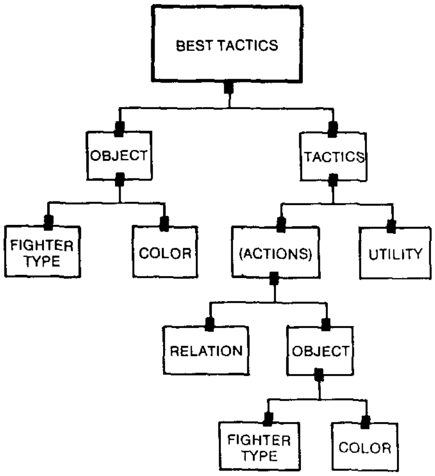


Fig. 10. Best tactics data structure.

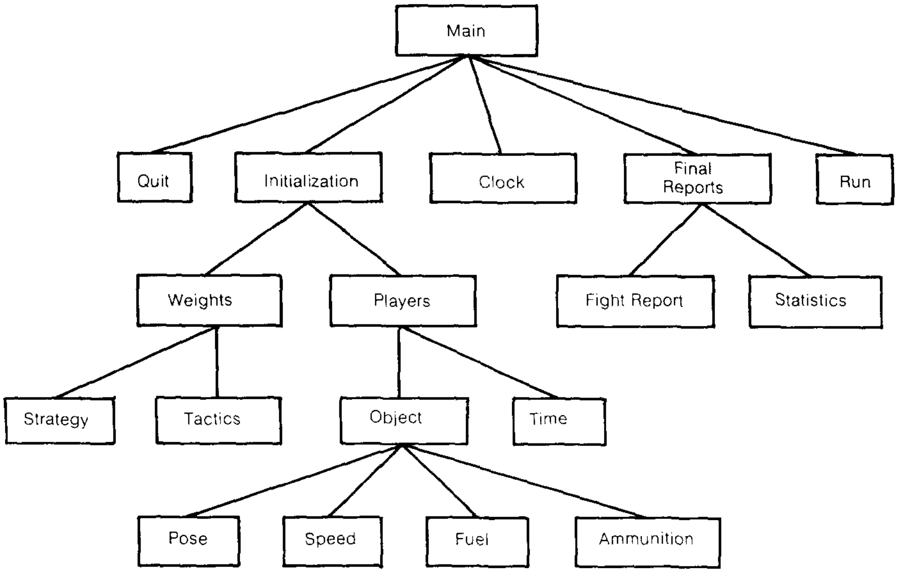


Fig. 11. Main menu system.

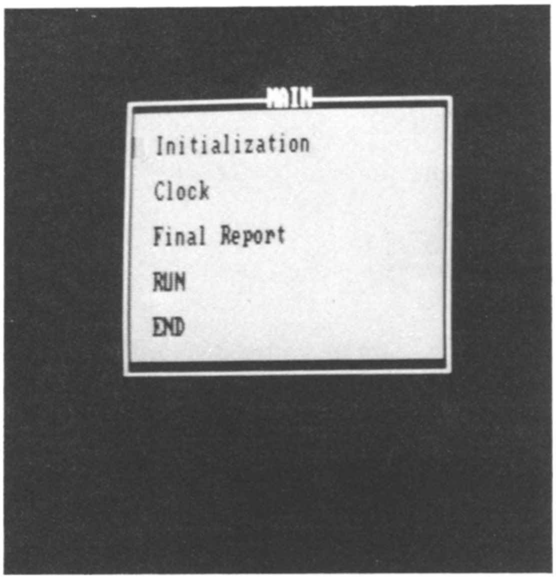


Fig. 12. Main menu.

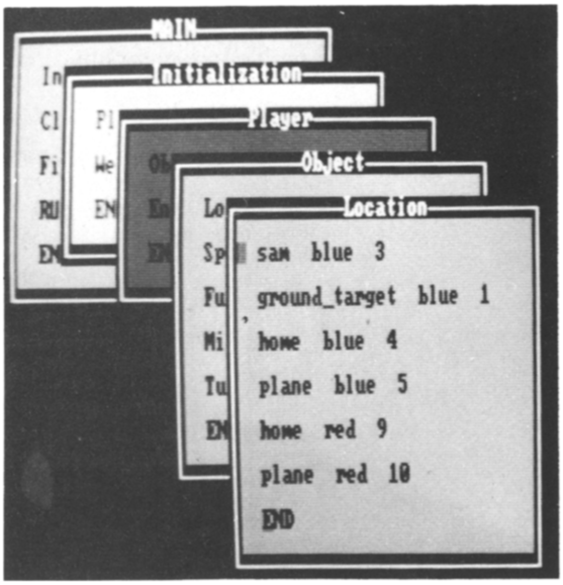


Fig. 13. Initialization menus.

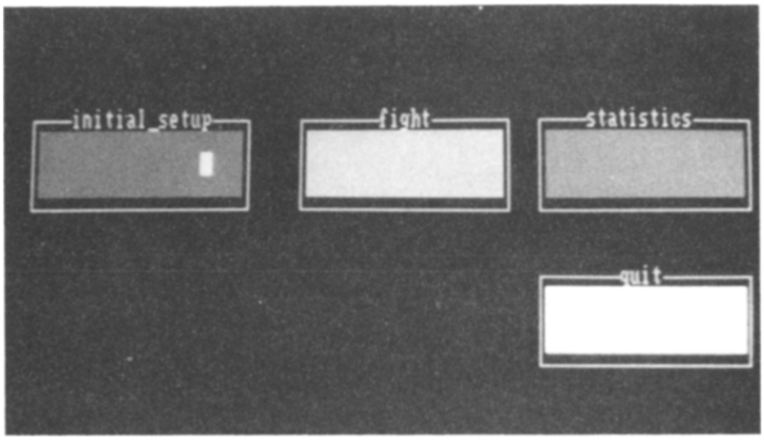


Fig. 14. Debriefing menus A.

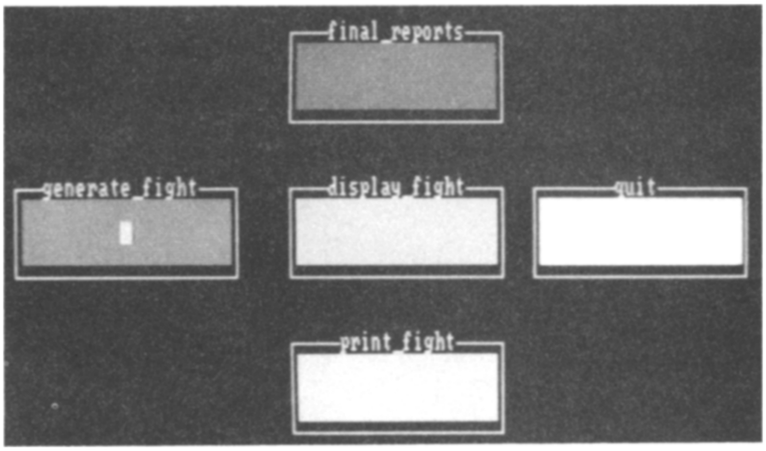


Fig. 15. Debriefing menus B.

In Figs 16 and 17 we are showing two illustrations of debriefing.

Finally, in Figs 18–23 we demonstrate a simulated game.

3. CONCLUDING REMARKS

This paper proposes the use of an Expert System in conjunction with pursuit–evasion algorithms and a pilot-derived Expert Data Base, in order to valuate alternative strategies for air combat. The most promising alternatives are communicated to the pilot allowing him to select the appropriate

```
INITIAL SETUP

      Player         I.D.  Weight    Location    Speed  Fuel  Timer
blue SAM             3       10.0  20000, 20000                4.8
  missiles:          g.a.  20  20  20  10  10
blue ground target   1      100.0  25000, 20000                0.5
blue home            4        0.0  25000, 25000                4.6
blue plane           1       50.0  25000, 25000  55.0   200    4.6
  missiles:          a.a.  30  30  20  10  10
                     a.g.  50  30  0
red home             9        0.0  14000, 10000                3.9
red plane            10     100.0  14000, 10000  95.0   200    3.9
  missiles:          a.a.  60  40  30  20  10
                     a.g.  50  50  30  30
-------------------------------------------------------------------
color                   survive      destroy
                        weights      weights
                          w_s          w_d

red                     100.00        0.00
blue                     20.00       80.00
```

Fig. 16. Initial setup report.

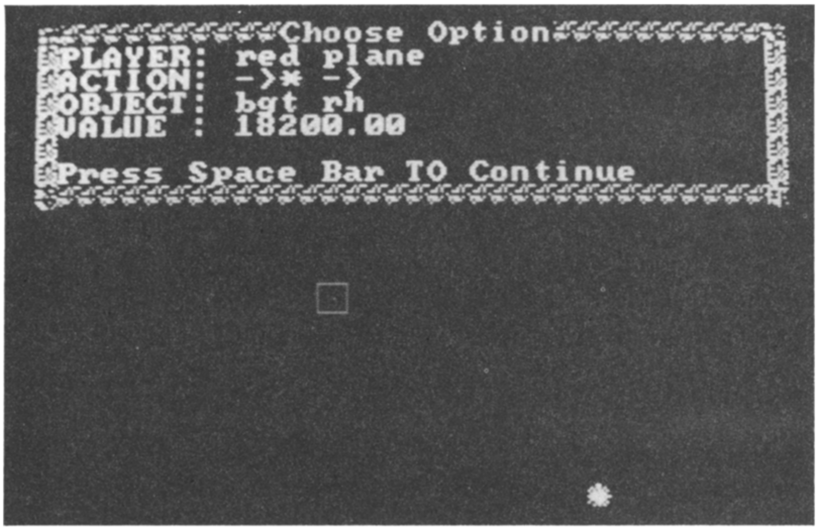
FLIGHT REPORT ORDERED ON TIME

Time	Destroyed		By		Where
4.80	blue ground target	1	red plane	10	25000, 20000
4.85	blue plane	1	red plane	10	22447, 23995
5.10	blue SAM	3	red plane	10	20000, 20000

SHOOTING REPORT

Time	Player	I.D.	Target	I.D.
4.80	red plane	10	blue ground target	1
4.81	blue SAM	3	red plane	10
4.85	red plane	10	blue plane	1
4.91	blue SAM	3	red plane	10
5.01	blue SAM	3	red plane	10
5.10	red plane	10	blue SAM	3

Fig. 17. Fight report.



Choose Option
PLAYER: red plane
ACTION: ->* ->
OBJECT: bgt rh
VALUE : 18200.00

Press Space Bar TO Continue

Fig. 18. Red plane (in square) receives its first mission to pursue the ground target (star) tactics: line of sight.

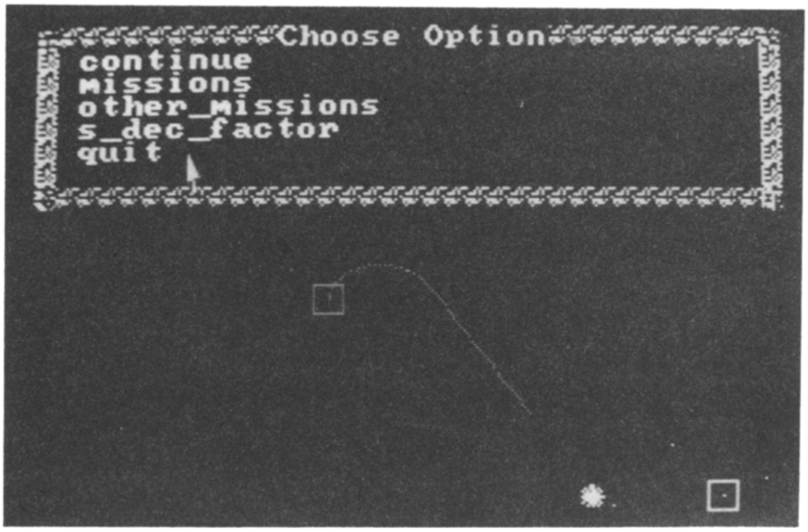


Fig. 19. New situation recognized: a blue plane (in square, bottom right) appears to defend the ground target.

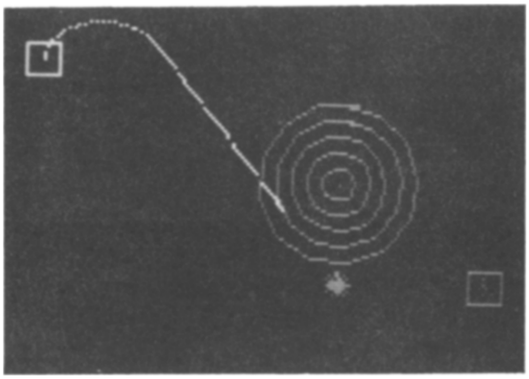


Fig. 20. Further new situation recognized: a blue SAM installation (concentric circles) is attacking the red plane.

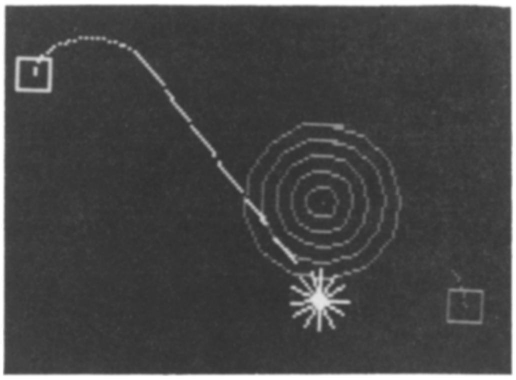


Fig. 21. Red plane destroys blue ground target. Note that blue plane began its maneuvers.

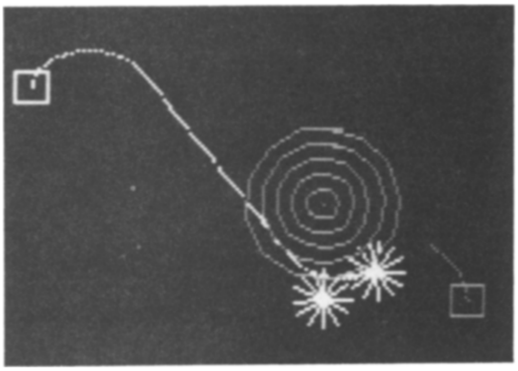


Fig. 22. Attacking red plane is destroyed by blue plane. Game ends.

action. The general framework for such a system is discussed in some detail and an implementation on a personal computer using the PROLOG language is described. The results of this study show that Artificial Intelligence and in particular the Expert System approach in conjunction with advanced computers and pursuit and evasion algorithms may hold great promise for the realization of a capable and reliable pilot decision aiding system for Flight and Fire Control System applications.

Finally, we mention that in addition to the aspects described on the previous pages, the following areas are currently under our investigation:

(1) a methodology for using the previous fighting experience in the Expert Systems,
(2) a methodology for evaluating airborne expert systems,
(3) the expansion of the current Knowledge Base and Inference Engine in order to deal with 3-dimensional geometry,
(4) the design of expert systems which effectively cooperate with each other,
(5) a complete and nonambiguous language for mission specification,
(6) the use of parallel computer architecture,
(7) heuristics determination to allow efficient mission generation.

REFERENCES

1. Automation in Air Combat—Committee on Automation in Combat Aircraft, Air Force Studies Board, Assembly of Engineering, National Research Council Publication National Academy Press, Washington, D.C. (1982).
2. E. Y. Rodin, Y. Lirov, S. Mittnik, B. McElhany and L. Wilbur, Pursuit and evasion by artificial intelligence. Presented at the *Second International Symposium on Differential Games Application*, Williamsburg, Va (1986).
3. C. V. Negoita, *Management Applications of System Theory*. Birkhauser, Basel (1979).
4. N. J. Nilsson, *Principles of Artificial Intelligence*. Tioga, Palo Alto (1980).
5. *Corporate Trackfile User's Manual*, J. Vinluan, Rockwell-NAAO, TFD 1849 (1985).
6. R. Isaacs, *Differential Games*. Wiley, New York (1965).
7. S. Mittnik, Hierarchical development planning with feedback. *Proc. 1st Latin American and 5th Brazilian Congress on Automatic Control* (1984).