Situation Assessment in the Paladin Tactical Decision Generation System

John W. McManus
NASA Langley Research Center
Hampton, Virginia

Alan R. Chappell
Lockheed Engineering and Sciences Company
Hampton, Virginia

P. Douglas Arbuckle
NASA Langley Research Center
Hampton, Virginia

SUMMARY

Paladin is a tactical decision generation system for air combat engagements. Paladin uses highly specialized knowledge–based systems and other Artificial Intelligence (AI) programming techniques to addresses the modern air combat environment and agile aircraft in a clear and concise manner. Paladin is designed to provide insight into both the tactical benefits and the costs of enhanced agility. The system was developed using the Lisp programming language on a specialized AI workstation. Paladin utilizes a set of air combat rules, an active throttle controller, and a situation assessment module that have been implemented as a set of highly specialized knowledge–based systems. The situation assessment module was developed to determine the tactical mode of operation (aggressive, defensive, neutral, evasive, or disengagement) used by Paladin at each decision point in the air combat engagement. Paladin uses the situation assessment module and the situationally dependent modes of operation to more accurately represent the complex decision-making process of human pilots. This allows Paladin to adapt its tactics to the current situation and improves system performance. This paper discusses the details of Paladin's situation assessment and modes of operation. The results of simulation testing showing the error introduced into the situation assessment module due to estimation errors in positional and geometric data for the opponent aircraft are presented. Implementation issues for real–time performance are discussed and several solutions are presented, including Paladin's use of an inference engine designed for real–time execution.

1 INTRODUCTION

A modern and realistic air combat simulation that can be used to evaluate current and future air combat environments must have an intelligent system to select the combat maneuvers to perform throughout an engagement, called a Tactical Decision Generator (TDG) , and the ability to model agile aircraft. The system should have a modular software structure so that the system can easily support modifications such as: new weapons systems or aircraft subsystems (e.g. sensors or propulsion systems), modifications to aircraft control systems, or changes to the aircraft's configuration. In support of the study of superagile aircraft at Langley Research Center (LaRC), a Tactical Guidance Research and Evaluation System (TiGRES) is being developed.[1,2,3]

The TiGRES system is designed to allow researchers to develop and evaluate aircraft systems in a tactical environment. The three main components of TiGRES are a TDG, the Tactical Maneuver Simulator (TMS), and the Differential Maneuvering Simulator (DMS). Both the TMS and the DMS use a TDG as the intelligent automated opponent.

The TMS[1,3] provides a high-fidelity batch air combat simulation environment for the development and testing of various guidance and control strategies. The researcher defines the initial conditions of the air combat engagement and the TMS then controls the trajectories and attitudes of the aircraft using simple trajectory commands, or through a tactical decision generation system. The main elements of the TMS are a high-fidelity, nonlinear six degree-of-freedom (d.o.f.) rigid-body aircraft dynamic model, including the control system, a TDG, and a user interface.

The DMS consists of two 40' diameter domes and one 20' diameter dome. The facility is intended for the real-time simulation of air combat engagements between piloted aircraft. By using a TDG to control one of the airplanes, it is possible to test a TDG against a human opponent. This feature allows the guidance logic to be evaluated against one or more unpredictable and adaptive human opponents.

While TiGRES is aimed specifically at the development and evaluation of tactical maneuvering strategies and advanced guidance/control systems for superagile aircraft, the modular design of TiGRES will

make it easily adaptable to the analysis of other aircraft systems (e.g. advanced weapons systems, advanced avionics systems).

2 THE PALADIN SOFTWARE

Paladin is a knowledge-based TDG designed to provide researchers insight into both the tactical benefits and the costs of superagility. Knowledge-based systems use a large amount of information about a problem's domain to understand and solve that problem. Paladin was developed in the Lisp programming language using a Symbolics 3650† workstation.

The development of Paladin has been a multi-stage process using a baseline version of the Adaptive Maneuvering Logic [4] (AML) program as the starting point. Paladin uses the trial maneuver generation and evaluation concept outlined in the AML program [4] with several extensions. The original set of five to nine aircraft trial maneuvers used by AML has been replaced with four sets of positionally dependent trial maneuvers. Past research [2,3] has shown that the use of the positionally dependent sets of trial maneuvers improves overall system performance, allows Paladin to perform target acquisition and tracking more effectively, and improves Paladin's defensive and evasive maneuvering performance. Paladin uses an object-oriented programming approach [5] to represent each aircraft in the simulation. Each aircraft object includes information on the current state of the aircraft's offensive systems (e. g. Guns, missile systems, fire control radars, ect.), defensive systems (e.g. electronic counter measures, chaff, ect.), and propulsion system. This state information is used to help guide Paladin's reasoning process.

Paladin was designed to utilize modular software separate subroutines and specialized computer hardware. The separation of the aircraft simulation and decision logic components, and the use of highly specialized knowledge sources, allows each module or knowledge source to be designed and implemented using the hardware and programming techniques specifically suited for its function. The use of highly specialized and independent knowledge sources also provides for "modular protection"[5], confining the effect of an error occurring in a module at run-time to that module, or to a small set of neighboring modules in the program. The confining effect of the modular protection was used to aid in the design and debugging process. Each knowledge source was developed and tested independently before it was incorporated into Paladin.

† Symbolics 3650 is a registered trademark of Symbolics Incorporated

The independence of the knowledge sources also increases the efficiency of Paladin by allowing knowledge sources to be distributed across a network of several heterogeneous processors. The network currently consists of a Symbolics 3650 workstation, a Symbolics MacIvory† workstation, two SUN† SPARC class workstations, and several Vax 3200† class workstations. Communication between the distributed knowledge sources is achieved using customized DECNet–based Client/Server software developed in–house for TiGRES. This software allows for synchronization, communications, and data sharing between heterogeneous computers running the DECNet communications protocol. TCP/IP based communications software has also been developed in–house for the SUN workstations. Paladin is implemented as a serial blackboard system, so no serialization or concurrency related software is required[6]. Each knowledge source requests all of the data required to perform its computation from the blackboard at the start of its execution cycle, and posts its results to the blackboard at the end of its execution cycle.

2.1 The Paladin Inference Engine

The Paladin knowledge sources use a custom inference engine (see appendix A) that was designed to support real-time execution of knowledge–based systems. The inference engine uses a depth–first evaluation strategy [5] to search the active rule–bases. Rule–bases can be partitioned, and the partitions can be linked using meta–rules (rules used to guide the activation of rule–bases). Rule–bases can be expressed in two formats: interpreted lists of condition action pairs, and compiled lists of in-line function definitions. The interpreted lists are used to develop and debug the initial versions of the rule–base. The rule–base is then "compiled" into a list of in–line functions. The rule–base compiler is written in Lisp and runs on an AI workstation. The compiled rule–bases execute approximately 90 to 100 times faster than the interpreted rules. The inference engine executes a representative test rule–base consisting of 40 rules in the interpreted format in 170 milliseconds. The inference engine executes the same rule–base in the compiled format in 1.9 milliseconds.

† MacIvory is a registered trademark of Symbolics Incorporated.

† SUN is a registered trademark of Sun Microsystems Incorporated.

† Vax 3200 & DECNet are registered trademarks of Digital Equipment Corporation.

It is important to realize that there is a trade–off between the length of the rule–base's longest execution path and the knowledge source's execution time. The shorter the execution path is, the faster the execution time. The rule–bases used by Paladin have been partitioned to increase system performance by grouping related rules into small partitions and using meta–rules to link the partitions. This partitioning decreases the number of rules that are active, and decreases the length of the worst–case execution path through the rule–base. The rule–base partitioning allows the designer to calculate the longest and shortest path through the rule–base and compute both a maximum and minimum knowledge source execution time. The knowledge source's maximum execution time can be used to insure that the system will meet real–time execution requirements. If the maximum execution time exceeds the allocated execution time, the designer may be able to repartition the rule–base until real–time execution requirements are achieved.

Paladin uses two rule–bases: a mode selection rule–base used by the Situation Assessment knowledge source, and a throttle control rule–base used by the Active Throttle Controller. The mode selection and throttle control rule–bases are included in appendices B and C. The mode selection rule–base consists of four partitions and contains nineteen rules. The shortest execution path in this rule–base results in a single rule being fired; the longest path results in twelve rules being fired. The throttle control rule–base consists of ten partitions and contains forty rules. The shortest execution path in this rule–base results in a two rules being fired; the longest path results in thirteen rules being fired.

2.2 Situation Assessment Module

Six modes of operation, shown in table 1, have been incorporated in Paladin. As shown in Figure 1, the Situation Assessment knowledge source is executed at the start of each decision interval, before the maneuver scoring module. The Situation Assessment knowledge source uses the mode selection rule–base (appendix B) to determine the system's current mode of operation. This knowledge source is used to model a pilot's situational awareness and changing problem–solving strategies. Just as a pilot will recognize the difference between an aggressive situation and a evasive situation and react accordingly, the Situation Assessment knowledge source provides information allowing Paladin to adapt its problem–solving strategy based on the current situation. The determination of the current mode of operation is based on the aircraft's current mission, the current state of the aircraft's systems, the relative geometry between the aircraft and its opponent, and the opponent's instantaneous-intent (defined later). Each of the six modes of operations has a unique vector of scoring weights and a unique decision interval (Δ sec). The scoring weights[2] for each mode of operation have been adjusted during the design and testing process to maximize Paladin's performance in that mode of operation. The testing procedures used to evaluate Paladin's performance is described in detail later in this paper.

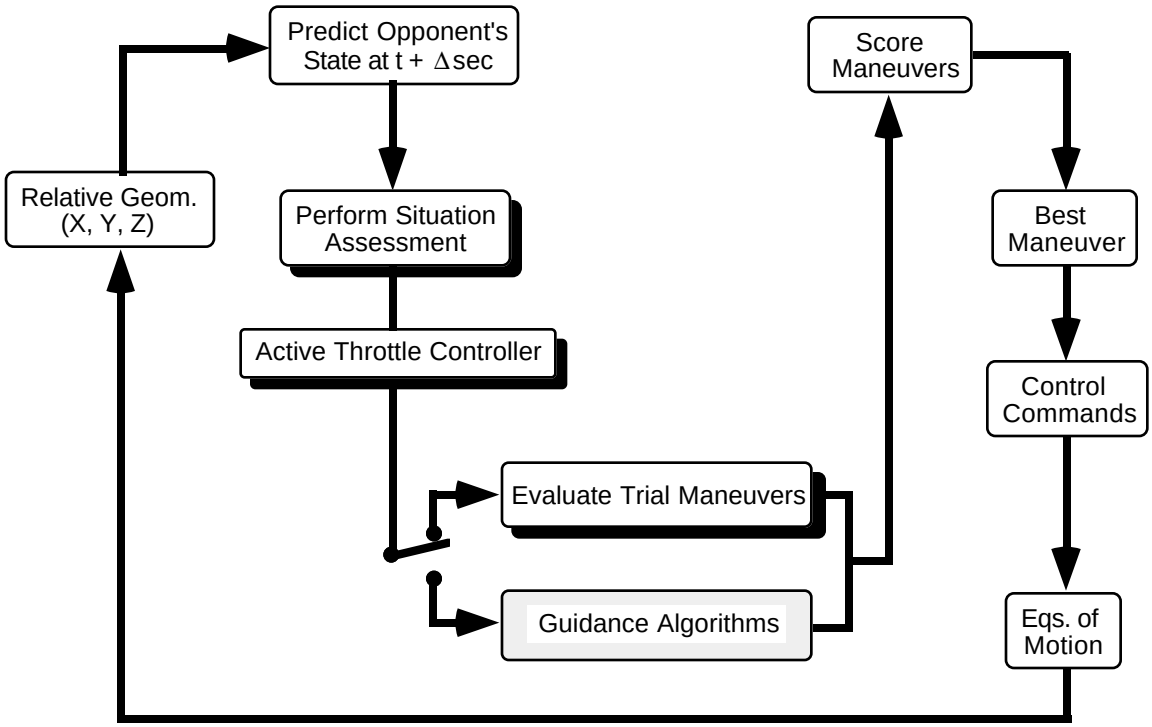

Figure 1. Schematic Of Paladin

Both TMS and DMS test results[1,2] have shown that a short decision interval (0.25 sec.) improves Paladin's fine–tracking performance in aggressive situations, and Paladins maneuvering capabilities in evasive situations. In neutral or defensive situations the same short decision interval results in a "thrashing" motion, degrading system performance. The thrashing is due to the system overcompensating for small changes in the opponent's motion. These thrashing maneuvers bleed off energy and reduce Paladin's effectiveness; thus a longer decision interval is used in defensive (0.5 sec.) and evasive situations (1.0 sec.).

Table 1. Modes of Operation

Mode	Decision Interval
Aggressive	0.25 sec
Defensive	0.5 sec
Evasive	0.25 sec
Ground Avoidance	0.125 sec
Neutral	1.0 sec
Disengage	0.5 sec

The situation assessment knowledge source also determines the opponents instantaneous-intent. The opponent's instantaneous-intent is defined to be an estimation of the opponent's mode of operation at the current point in time based on Paladin's available sensor, positional, and geometric data. Currently, there is no attempt to use a history of instantaneous-intent to derive a long-term opponent intent.

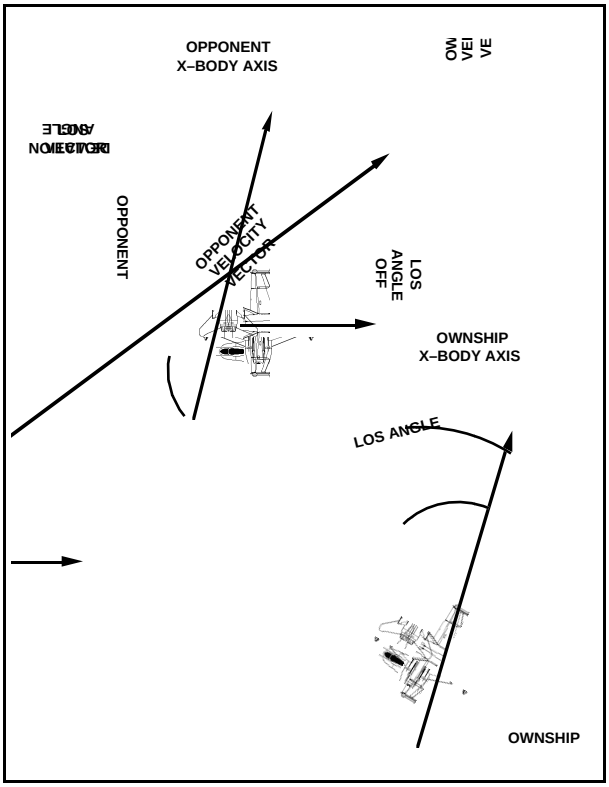

Figure 2. Angle Definitions

2.2.1 *Situation Assessment Data*

To perform situation assessment, information on the relative geometry between the two aircraft and Paladin's system status is required. This information is available in the form of participant-specific data maintained by Paladin. All data relating to the Paladin aircraft as well as Paladin sensor data (e.g. the opponent's relative position) are known exactly. Other data required about the opponent must be estimated.

The quantities used by the situation assessment module which are based on exactly known data are either specific to the Paladin aircraft or are relative values from the Paladin aircraft's point of view. Paladin's current throttle position and altitude are parameters taken directly from the current state. Range is the magnitude of a vector connecting the centers of gravity of the aircraft. The Line-Of-Sight (LOS) angle is defined as the angle between the LOS vector and the ownship body x-axis (figure 2); the deviation angle is defined as the angle between the LOS vector and the ownship velocity vector; and the LOS angle off is defined as the angle between the LOS vector and the opponent's body x-axis.

The deviation angle is calculated as the inverse cosine of the magnitude of the projection of the range into the velocity axis divided by the range. In equation form,

deviation angle =

$$\arccos\left[\frac{\dot{x}\Delta x + \dot{y}\Delta y + \dot{z}\Delta z}{(\text{Range})\,|\text{Velocity}|}\right], \tag{1}$$

The line-of-sight angle (LOS) is the inverse cosine of the magnitude of the projection of the range into the x-body axis divided by the range, or,

LOS angle =

$$\arccos\left[\frac{D(1,1)\Delta x + D(1,2)\Delta y + D(1,3)\Delta z}{\text{Range}}\right], \tag{2}$$

where Δx, Δy, and Δz represent the difference between the two aircraft positions. D(i,j) is the i, j element of the Paladin body axis direction cosine matrix. Then the LOS elevation is taken to be the inverse sine of the opponent's z-coordinate in the Paladin body axis system divided by the range, or,

LOS elevation =

$$\arcsin\left[\frac{-z_{\text{opponent in Paladin body axis system}}}{\text{Range}}\right]. \tag{3}$$

The LOS azimuth is the inverse tangent of the opponent's y-coordinate divided by the opponent's x-coordinate, both in the Paladin body axis system,

LOS azimuth =

$$\arctan\left[\frac{y_{\text{opponent in Paladin body axis system}}}{x_{\text{opponent in Paladin body axis system}}}\right]. \tag{4}$$

Finally, off corner is the proportion by which Mach number differs from the instantaneous cornering Mach number (speed to achieve largest possible turn rate) calculated for the current altitude,

off corner =

$$\frac{\text{Cornering Mach - Current Mach}}{\text{Cornering Mach}}. \tag{5}$$

The velocity / acceleration and orientation of the opponent must be estimated, since this data would not be available from realistic Paladin sensors. These estimates are made using a three point time history of the known position data and several assumptions about the opponent aircraft's aerodynamics (weight, wing surface area, and flight characteristics). The current position of the opponent and the opponent's position at the preceding two decision cycles are used to find a quadratic equation for the position as a function of time. The first and second derivatives of this function at the current time yield an estimation of the opponent's instantaneous velocity and acceleration. By assuming the aerodynamic characteristics of the opposing aircraft, and using the velocity and acceleration estimates, an estimated body-axis orientation for the opponent can be found.

The quantities used by the situation assessment module which are based on estimated data are largely relative values from the opponent aircraft's point of view. Each of these quantities has some error introduced by the estimation process. The range rate is the magnitude of the projection of the relative velocity onto the range axis, or,

Table 2. Error Statistics

Range Rate Error (ft/sec)		Deviation Error (deg)		LOS Error (deg)	
$\bar{X}$	s	$\bar{X}$	s	$\bar{X}$	s
1.52	0.68	2.06	0.13	8.02	3.05

$$\text{range rate} = \frac{\Delta\dot{x}\Delta x + \Delta\dot{y}\Delta y + \Delta\dot{z}\Delta z}{\text{Range}}. \tag{6}$$

The opponent's deviation angle and line-of-sight angle are calculated similarly to the Paladin aircraft parameters (equations 1 and 2), using the opponent's velocity and x-body axis. Paladin's LOS angle off is 180° - opponent's LOS angle. The errors between the actual parameter values and the estimated values were calculated for a representative set of 32 within–visual-range engagements[3], resulting in the sample means ($\bar{X}$) and standard deviations (s) listed in table 2. Figures 3 through 5 show each of these error magnitudes (absolute value of the actual value minus the estimated value) during the course of a typical engagement. Error expectations given in table 2 and magnitudes given in the figures are based on engagements between Paladin and an opponent with known aerodynamic characteristics. If the aerodynamics of the opponent aircraft are not well known, the error in the LOS angle should increase, since the error is strongly dependent on the aircraft flight characteristics.

From the same set of 32 engagements, results were collected on the sensitivity of the situation assessment module to these estimation errors. The correct mode (the mode selected given exact data) of operation was chosen 98.0% of the time. Hence, this implementation of the situation assessment knowledge source is believed to be insensitive to the errors introduced by data estimation.

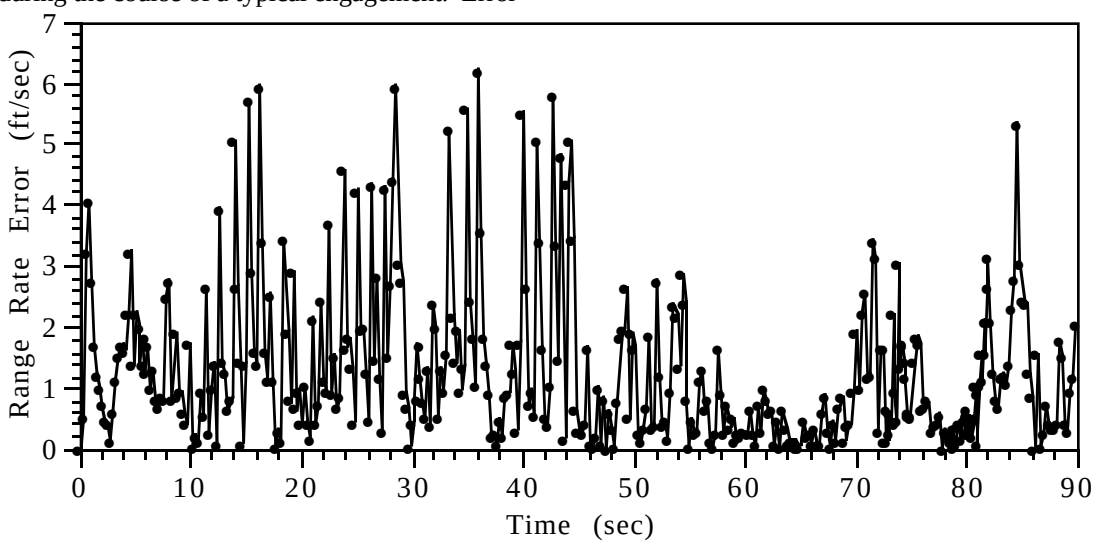

Figure 3 Range Rate Error during Engagement.

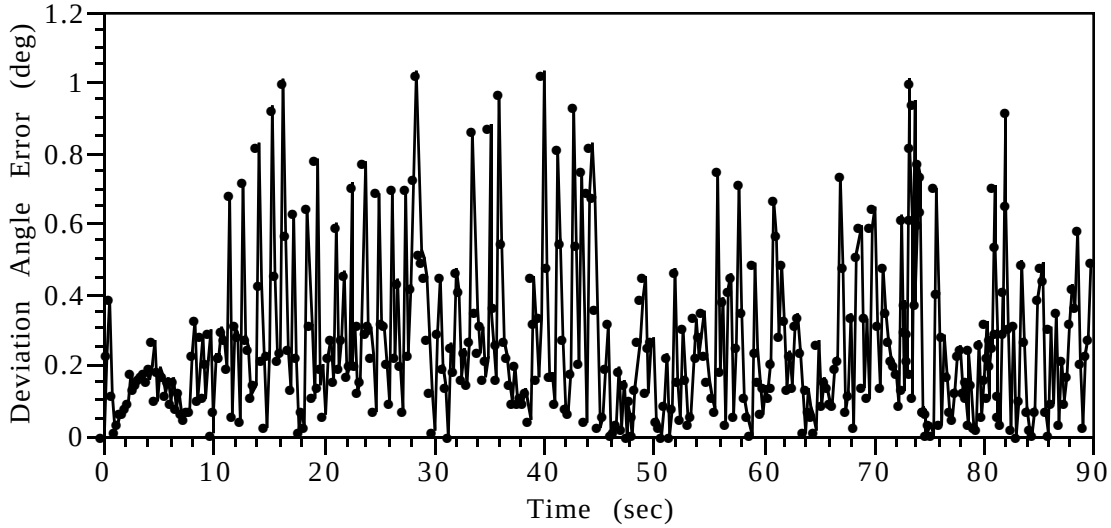

Figure 4 Deviation Angle Error during Engagement.

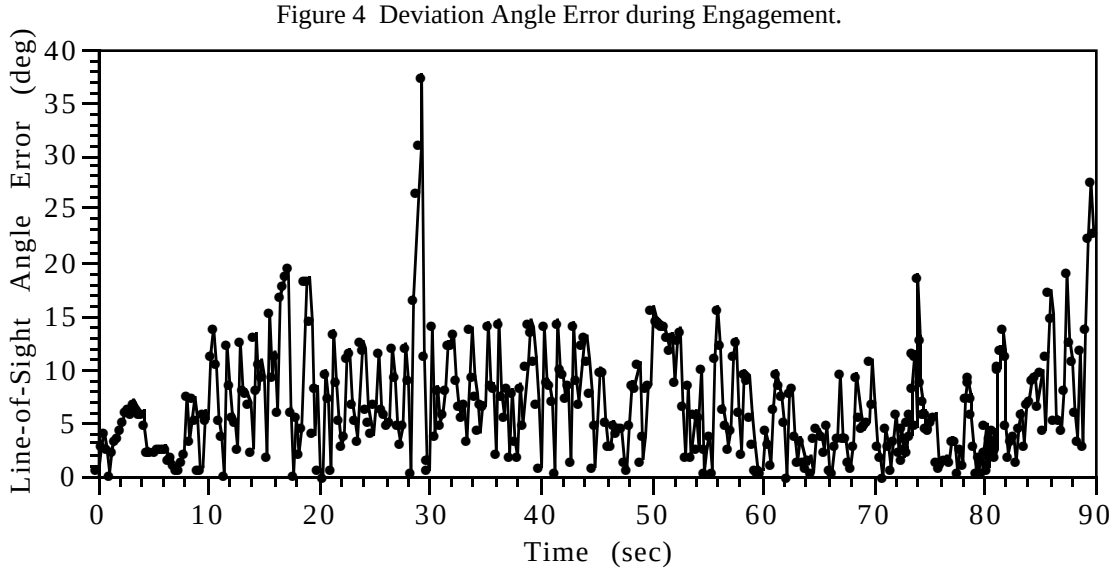

Figure 5 Line-of-Sight Angle Error during Engagement.

2.3 Active Throttle Controller

A rule-based Active Throttle Controller was developed to adjust the throttle setting based on the current mode of operation. The throttle controller is called at the start of each decision interval and can set the throttle to any position between idle and full afterburner [0.0 = flight idle, ..1.0 = military power, ..2.0 = full afterburner]. The throttle controller uses the throttle control rule–base (appendix C), the current mode of operation, and the relative geometry information to select either a target acquisition mode, a fine tracking mode, or a target or missile avoidance mode. Each mode has a set of specific throttle control rules that are used to maximize system performance in that mode.

The active throttle controller uses the same data described for the situation assessment module, and so, incurs the same estimation errors. Using the results of the 32 engagements discussed in the previous section, the resulting errors in the selected throttle position have been evaluated. The throttle setting chosen by the active throttle controller was within +/-5% of the correct setting (the position selected given exact data) in 95.8% of the cases tested (7369 total). Table 3 shows the distribution of these errors around the correct throttle command. For this table, E is the error band (in %) of the throttle position, and P is the percentage of the test cases which fall within +/- E of the correct position.

Table 3. Active Throttle Controller Error

E	P
5	95.81
10	95.83
15	95.90
20	97.15
50	99.38

2.4 Maneuver Scoring Module

The Paladin Maneuver Scoring Module knowledge source is a FORTRAN subroutine that uses a set of fuzzy logic questions with responses ranging from [-1.0 = Negative, ..0.0 = Neutral, ..1.0 = Positive] and the mode-specific scoring weight vector selected by the situation assessment module to score each of the trial maneuvers. For each trial maneuver evaluated the predicted positions for both the opponent and the Paladin aircraft are computed. The position of the opponent is extrapolated using a quadratic curve fit based on the time history of the opponent aircraft's trajectory as previously described. The future position of the Paladin aircraft is determined by predicting the result of executing the control commands for each candidate trial maneuver.

Once the relative geometry between the two future aircraft positions is calculated, the score for the maneuver is determined by computing the responses to the seventeen fuzzy logic questions, applying the selected scoring weight vector, and then summing the responses to generate a single numeric score. After all of the trial maneuvers have been evaluated, the highest scoring maneuver is selected and the associated control commands are executed.

3 Engagement Scoring Metrics

Four scoring metrics are currently used to evaluate each engagement. All metrics are computed at the aircraft simulation update rate of 32 times per second. The first metric computes the total time that each airplane has its weapons locked on its opponent, the probability that any weapons fired will hit the opponent, the distance between the opponents, the angle-off, and the deviation angle. The results are printed in a table format at the completion of each run.

The second scoring metric computes a Probability of Survival (PS) using the data computed by the first metric. The probability to hit for an all–aspect missile and for the cannon are computed using the range and LOS angle to the opponent. The probability to hit for a tail–aspect missile is computed using the range, the LOS angle to the opponent, and the LOS angle off. Aircraft missiles are treated as limited resources and a probability to hit of 0.65 is required to launch the first missile. The probability to hit threshold increases by 0.05 for each missile launched. An estimated flyout time (the time it will take a missile to reach it's target) for each missile is computed based on the launch parameters, and another missile cannot be fired until the flyout time has passed. The Ps for an aircraft then is

$$Ps = 1.0 - \Sigma\,[\text{probability to hit} * Ps(f)] \tag{7}$$

summing over each weapon fired by the opposing aircraft. Ps(f) represents the Ps of the aircraft firing the weapon at the time the weapon was fired.

The third scoring metric attempts to determine a Lethal Time (LT) advantage for each engagement. Lethal time advantage attempts to weigh the lethality of each distinct type of weapons lock time.

$$LT = \frac{\text{Paladin Gun - Opponent Gun}}{2} + (2 * (\text{Paladin Tail–Aspect - Opponent Tail–Aspect})) + (\text{Paladin All–Aspect - Opponent All–Aspect}) \tag{8}$$

A positive lethal time value shows Paladin with a lethal time advantage, and a negative lethal time shows the opponent with an advantage.

The fourth metric is Time on Offense (TOF).

$$TOF = (\text{Gun time} + \text{All-aspect time} + \text{Tail-aspect time}) \tag{9}$$

ΔTOF is computed as Paladin's TOF minus the opponent's TOF. As for LT, a positive ΔTOF value shows Paladin with an time on offense advantage, and a negative ΔTOF shows the opponent with a time on offense advantage.

4 Paladin Testing Procedures

Paladin is currently being tested in the TMS using six d.o.f. aircraft dynamics, and in the DMS using five d.o.f. aircraft dynamics. TMS testing is done in a non-real-time, batch mode environment against a baseline TDG. Each set of test conditions consists of 32 sets of initial aircraft conditions. The initial altitudes, airspeeds, and the separations between the two aircraft

are adjusted to provide representative coverage of the within–visual–range air combat arena. The largest initial aircraft separation currently being tested (5 nm) places the aircraft at the transition point between beyond-visual-range and within-visual-range air combat.

The scoring metrics discussed earlier are reviewed after each set of test runs and the data are used to tune the mode specific scoring weights and test the completeness of the knowledge bases. Although the metrics are helpful, no single metric has been developed that can completely measure the performance of an aircraft in the engagement. In past test engagements an aircraft could score significant amounts of weapons lock time after it had been "killed." This phenomenon adversely affected several of the scoring metrics. To correct this problem all engagements are now ended when the probability of survival for either aircraft is less than 0.30.

After initial adjustment of the scoring weights, the set of initial conditions is expanded to 320 initial conditions by modifying the initial separation between the airplanes, the initial altitudes, and the initial Mach numbers. This stepwise refinement process provides the large sets of results required to achieve global system improvements across the total within–visual–range air combat environment.

A baseline version of Paladin is currently being tested in the DMS using a 5 d.o.f. aircraft model. The aircraft model lacks both the extra degree of freedom (side force) as well as an accurate representation of the aircraft's rotational dynamics throughout the complete flight envelope. The baseline Paladin system, the Computerized Logic for Air Warfare Simulation (CLAWS), is a blackboard system that contains the situation assessment module, the active throttle controller, and a reduced set of situationally dependent trial maneuvers. This reduced set of trial maneuvers and the simplified aircraft model are used to insure real-time performance in the DMS.

The development of CLAWS has made it possible to evaluate the tactical decision generation software against human pilots in a realistic air combat environment. This capability has allowed experienced pilots to interact with the system and comment on its performance and suggest improvements. The pilots' comments and suggestions are then the basis for changing the TMS experimental version of Paladin. These changes are tested and refined before being included in the baseline system.

5 Concluding Remarks

Paladin has been developed to study within-visual-range air combat engagements. The system incorporates modern airplane simulation techniques, sensors, and weapons systems. The system was developed using several concepts first outlined in the Adaptive Maneuver Logic program. The use of knowledge–based systems and Artificial Intelligence (AI) programming techniques allows Paladin to address air–to–air combat and agile aircraft in a clear and concise manner. The ability to integrate Paladin into the Differential Maneuvering Simulator offers a unique opportunity to evaluate the performance of the AI-based Paladin software in a real-time tactical environment against human pilots.

The knowledge–based Situation Assessment module and the Active Throttle Controller allow Paladin to function in the complex modern air–to–air combat environment. Paladin is able to model aspects of the complex decision making–processes used by human pilots through the use of the Situation Assessment knowledge–based system. The use of distinct modes of operation allows Paladin to model complex air–to–air combat tasks and generate tactical decisions in real–time. Paladin presents an excellent opportunity to evaluate the use of AI programming techniques and knowledge-based systems in a real-time environment.

References

1. Goodrich, Kenneth H; McManus John W. : *"Development of A Tactical Guidance Research and Evaluation System (TiGRES)."* AIAA Paper #89-3312, August 1989.

2. McManus John W.; Goodrich, Kenneth H. : *"Application of Artificial Intelligence (AI) Programming Techniques to Tactical Guidance for Fighter Aircraft."* AIAA Paper #89-3525, August 1989.

3. Goodrich, Kenneth H; McManus John W. : *"An Integrated Environment For Tactical Guidance Research and Evaluation."* AIAA Paper #90-1287 May 1990.

4. Burgin, G. H., et al.: *An Adaptive Maneuvering Logic Computer Program for the Simulation of One-on-One Air-to-Air Combat.* Vol I and Vol II. NASA CR-2582, CR-2583, 1975

5. Meyer, Bertrand. *Object-oriented Software Construction.* Ed. C.A.R. Hoare. Prentice Hall International Ltd, 1988.

6. McManus John W.: "A Parallel Distributed System for Aircraft Tactical Decision Generation" In *Proceedings of the 9th Digital Avionics Systems Conference*, 1990, pp. 505 – 512