An Experiment Adapting Adversarial Game Trees to Computer Generated Forces in Real Time, Dynamic, Asynchronous Environments

1st Lt. James Benslay

USSTRATCOM/J621
901 SAC Blvd, Suite 2B10
Offutt AFB, NE 68113-6600
benslayj@stratcom.af.mil

Capt. Vincent Brian Zurita

ACC CSS/SCFR
37 Elm St., Room 113
Langley AFB, VA 23665
zuritav@acccss.langley.af.mil

Abstract

A game tree is described that operates in real time within a distributed virtual environment. Such an environment demonstrates asynchronous, highly dynamic qualities which are typically beyond the realm of adversarial game trees. The purpose of this game tree is to rapidly determine the best fighter aircraft maneuver to apply in a given situation involving the subject computer generated aircraft and a single opponent aircraft. The game tree uses fuzzy logic to describe and evaluate the spatial relationship between the two entities. A knowledge base of maneuvers is represented as matrices of three-dimensional points in space. Real time decisions are achieved by calculating a maximum allowable decision time which is dynamically constrained based on distance between entities. A variable look-ahead value models a variety of pilot performance capabilities. This game tree has been successfully tested in a subset of spatial combinations describing the positions and orientations of two opposing aircraft in a distributed virtual environment.

Introduction

Game trees are useful when used in an environment where possible moves between opponents are known and resulting positions can be enumerated. The resulting positions can then be weighed, and appropriate decisions made based on available paths through the tree that result in the most advantageous weighting. This is otherwise known as the mini-max procedure in game trees (1). Game trees are not naturally applied however, to situations where resulting positions between players are anything other than discrete. To do so results in a combinatorial explosion in the branching factor of the tree. However, if certain heuristics could be applied to the sets of otherwise non-discrete possible player positions, thus describing the positions in discrete ways, then it would be possible to limit the branching factor of a game tree and thus make it usable in this scenario. Such an approach was applied to the Intelligent Wingman, an experimental computer generated force (CGF) assembled at the Air Force Institute of Technology's Virtual Environments Laboratory.

Why even attempt to adapt the game tree concept to such a dynamic environment? The reason was that a method was needed to compare and evaluate potential maneuvering decisions with regards to concurrent adversarial maneuvers over the course of time. The game tree strategy of weighing potential outcomes from intermediate decisions along a course of action seemed the appropriate, initial approach to take. This decision was bolstered by the result of Schaper et al's (2) research that showed the feasibility of using adversarial game trees in a real time context. What was needed was to expand this concept to include asynchronous movement in a time constrained environment.

Approach

The implementation of the asynchronous game tree was an integral part of our earlier work on the Intelligent Wingman (3, 4). During our design, we identified the game tree algorithm as the core process of the maneuver decision module within the mission-level decision engine (previously named the tactical decision engine) of the Intelligent Wingman's architecture. See our previous work (3, 4) as well as Santos et al (5) for a detailed description of a generic CGF architecture. The algorithm development was completed in the following steps.

Step 1: Creating a Knowledge Representation

A CGF domain independent knowledge representation had to be developed to represent possible player moves. This was necessary to allow the knowledge to be abstracted from the algorithm used to act on the knowledge, so that the same game tree algorithm could be used for any CGF

regardless of the CGF type. We chose to build the moves, or maneuvers, as simplistic directional graphs within a three coordinate space. See Zurita (3) for an in-depth discussion of this process. One key advantage to representing the maneuvers in this three coordinate space is that the fundamental shape of the maneuver could be rotated and scaled using standard formulas to achieve any desired permutation of the basic maneuver. The representation was then described in a simple, six columnar format within a plain text file. The six columns of the file were x, y, and z coordinates, and the values for the CGF's heading, pitch, and roll at each of the points. Given these six attributes, it is possible to describe the location of a CGF with regards to the coordinate origin, as well as the intended orientation of the entity. By taking into account an entity's velocity vectors, applying a specific maneuver representation, and then rotating and scaling the maneuver by functions derived by the velocity vectors, we deduced that we would be able to predict the location and orientation of the CGF in a specified amount of time.

Step 2: Determining Maximum Decision Time

Given that we were using this approach to model maneuver decision making in the highly dynamic and time critical domain of combat aircraft operations, we needed a mechanism to determine the maximum amount of time that the game tree algorithm could use while searching for the best possible solution. By doing so, the game tree algorithm would never put the CGF entity in jeopardy by taking too long to make a decision. During our incremental approach to the Intelligent Wingman experiment, we were limiting ourselves to offensive maneuvers, i.e. those types of maneuvers used when the Wingman is pointed more or less towards the target aircraft, and the target aircraft is more or less pointed away from the Wingman. For these situations, we calculated the minimum time it would take the Wingman to intercept the target aircraft using fictitious data in place of classified missile flight parameters. This minimum intercept time was then divided by a user defined value, the result being the maximum allowable time the game tree algorithm could use to make a decision. The reason for adding this user defined divisor was to prevent the algorithm from using the entire time to intercept to make a decision. By giving the user control of this value, it allows flexible scaling of the performance factors in the CGF's decision making. In addition to the division value, an absolute minimum timing value is also available to the user in order to specify that should the resulting maximum decision time be very short (less than 1 second), the algorithm may over ride its calculation and use the minimum specified decision time.

For example, during our initial experiments, the Wingman and a bandit were started at 20 miles apart on an offensive intercept. Using the aircraft velocities and our fictitious missile data, an intercept time of approximately 400 seconds was calculated. We used a division factor of 100, thereby limiting the algorithm's maximum decision time to 4 seconds (which is reasonable according to fighter pilots we interviewed). The absolute minimum time was set to 1 second.

Step 3: Constructing the Game Tree

The actual game tree was constructed by using the bandit's known maneuvers for a defensive approach as the first tier nodes, and then combining all the wingman's maneuvers for an offensive approach as the second tier nodes. As a result of this approach, the ply of the basic tree will never exceed 2 (not including the depth of the evaluation branches discussed in Step 6).

Since the algorithm was written to accept any number of maneuvers when constructing the tree, the names of the known maneuvers as well as their data files were established at program startup by an initialization file. To keep the algorithm complexity low, we elected to not accept dynamically defined maneuvers after program start. Also during our initial tests, we assumed that the "pilots" of both CGFs would have equal maneuvering knowledge, hence we used the same set of offensive and defensive maneuvers for each CGF.

As a component of the maneuver nodes of each tier, the maneuver points that represented that maneuver were attached. This would be a necessary step in determining the location of each entity along their particular maneuver. Refer to Figure 1 for an example of the modified tree.

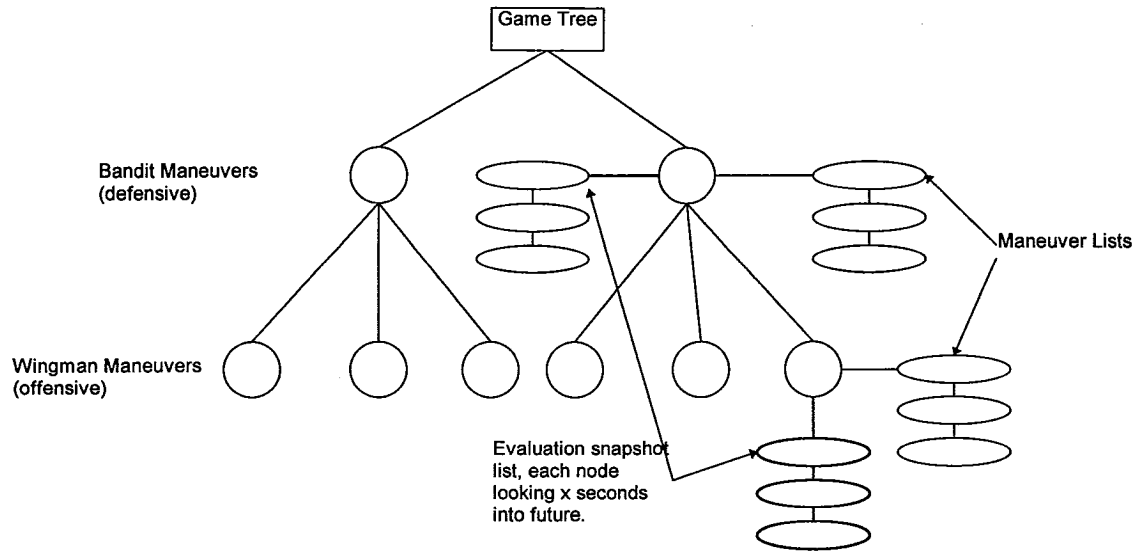

Figure 1. An example of our asynchronous game tree, showing lists for maneuver points and lists for evaluation snapshots.

Step 4: Determining Discrete Look-Ahead Values

It was an initial requirement that the Intelligent Wingman's performance capabilities be completely scaleable (6). To this end, fuzzy logic term sets, membership functions, and inferencing were used to assign imprecise descriptions of the Wingman's abilities (3, 4, 5, 6). The game tree algorithm then used the values of certain variables defined over fuzzy logic term sets as iterative multipliers to determine how far ahead that particular Wingman instantiation would be able to accurately predict both his and the bandit's position and orientation given a pair of chosen maneuvers. For our tests, the evaluations of the fuzzy values resulted in look-ahead values of 1, 2 or 3 seconds. What this means is that for each breadth wise pass through the game tree, a Wingman with a look-ahead value of 3 seconds would get to look 3 more seconds into the future of the maneuver combination. Hence, after 2 passes of the tree, the Wingman with a look-ahead of 3 would be 6 seconds into the maneuver, whereas the Wingman with a look-ahead of 1 would only be 2 seconds into the maneuver. Thus we were able to establish better foresight performance for those Wingman who had higher performance ratings.

Step 5: Determining Discrete Player Positions

By knowing the combination of chosen maneuvers and the aircraft velocities, we were able to create "snapshots" of player positions and orientations with regards to time as calculated by the product of the look-ahead value and the number of passes through the game tree. These snapshots were then used to represent the discrete player positions necessary to plug into the evaluation nodes of the game tree. As it turned out, understanding and implementing the multiple coordinate system transformations in preparation of evaluating the resulting positions ended up being a large portion of our effort. Without paying careful attention to the order and type of matrix transformations required, any resulting position evaluations would of course be meaningless. Refer to Zurita (3) for a detailed description of the required coordinate system transformations.

Step 6: Evaluating Resulting Player Positions

It is beyond the scope of this paper to describe in detail our process for identifying and implementing the fuzzy logic term sets and membership functions that we used to evaluate the player positional snapshots. Suffice it to say that we interviewed available fighter pilots and thereby defined the following term sets:

1) Relative Target Latitude,
2) Relative Target Longitude,
3) Relative Target Altitude,
4) Range to Designated Target,
5) Target Aspect, and
6) Firing Range.

We then created a reasoning schema (4) to outline the fuzzy inferencing rules necessary to affect the Evaluation term set.

Moving breadth-wise through the game tree, player positions were evaluated by one maneuver pairing at a

time, so long as there was both a combination left to evaluate and sufficient time remained to make the evaluation. A single evaluation involved fuzzifying the exact coordinate locations according to the previously mentioned term sets, and then allowing the inferencing engine to evaluate all the applicable rules. We used a standard, center of gravity defuzzification technique to derive a single, quantifiable result from the inferencing engine. This value was used as the resulting player positional score.

Experiment Results

In order to test our algorithm, we built in extensive data reporting utilities that would allow us to examine all aspects of the game tree construction and evaluation. Below is a simplified example of output data from one experiment. It details, in the following order: 1) pilot profile information, 2) look ahead calculations, 3) both Wingman and bandit initial locations and resulting orientations from the opposite perspective, 4) orientation calculations for fuzzification, 5) decision timing criteria, and 6) the resulting evaluations and a concluding decision maneuver.

GAME TREE DUMP DATA	
PP Membership Section:	
Fact Assertion & Inferencing Time:	0.044459
PP:	92
PP Membership in 'Weak':	0
PP Membership in 'Average':	0
PP Membership in 'Strong':	0.2
PP Membership in 'Elite':	0.2
Determine Lookahead Section:	
Calculation Time:	4.4e-05
TimeStep:	3
Acft Transformation Section:	
(Units: Feet & Degrees)	
FW_Pos.x:	0
FW_Pos.y:	0
FW_Pos.z:	2000
FW_Pos.psi:	5
FW_Pos.theta:	10
FW_Pos.phi:	0
FW_Pos.linear_velocity.x:	475
FW_Pos.linear_velocity.y:	475
FW_Pos.linear_velocity.z:	0
Bandit_Pos.x:	75000
Bandit_Pos.y:	2000
Bandit_Pos.z:	2000
Bandit_Pos.psi:	5
Bandit_Pos.theta:	0
Bandit_Pos.phi:	0
Bandit_linear_velocity.x:	450
Bandit_linear_velocity.y:	50
Bandit_linear_velocity.z:	0
FW_Trans_Pos.x:	74888.9
FW_Trans_Pos.y:	4544.29
FW_Trans_Pos.z:	0
FW_Trans_Pos.psi:	0
FW_Trans_Pos.theta:	0
FW_Trans_Pos.phi:	0
Bandit_Trans_Pos.x:	73753.8
Bandit_Trans_Pos.y:	4444.99
Bandit_Trans_Pos.z:	13023.6
Bandit_Trans_Pos.psi:	0
Bandit_Trans_Pos.theta:	0
Bandit_Trans_Pos.phi:	0
Determine BFM Geometry Section:	
Fact Assertion & Inferencing Time:	0.053217
RTLO:	79.4311
m(on_the_nose_of):	0.918379
m(even_with):	0
RTLA:	3.3965
RTA:	9.99641
m(even_with):	0.909124
TAS:	176.528
m(hot):	0
FIR:	75026.7
Score:	94.6746
m(advantages):	0.986687
BFM Geometry State:	Offensive
Tree Construction Section	
Wingman Posture State:	Offensive
Bandit Posture State:	Defensive
Decision Time Section:	
Units: Time(seconds), Distance(feet):	
Time to Calculate Decision Time:	9.3e-05
Distance to Bandit:	75026.7
Weapon Employment Zone (hard coded for now):	58080
Min Allowable Decision Time (via parameter):	1
Time to Weapon Optimum Range:	466.588
Total Time Division Factor (via parameter):	100
Max Allowable Decision Time (via calc):	4.66588
Total Evaluation Time:	3.11382
Scoring Only Time:	2.98858
Total Maneuver Nodes Considered:	312
Longest Single Evaluation Time:	0.065952
Highest Score:	94.8417
Chosen Wingman Maneuver	bfm.Close

Shortfalls of the Asynchronous Game Tree

To limit the scope of our effort, the game tree currently models only offensive maneuvering situations. This obviously falls short of modeling a real world environment in which opposing CGFs may also encounter each other in high aspect or defensive maneuvering settings. Incorporating high aspect and defensive maneuvers into the knowledge base will allow the asynchronous game tree to more accurately model an actual combat environment.

Additionally, there exists no separate knowledge base of maneuvers appropriate for a true adversarial CGF. We used the same offensive and defensive F-15 maneuvers for both the Intelligent Wingman as well as its foe. By developing a set of maneuvers that models a potential adversarial CGF (i.e. a MiG-29), this shortfall may be overcome.

Finally, the current algorithm doesn't employ a true mini-max procedure in that it doesn't maximize the adversary's move. By allowing such a maximization, the mini-max procedure may be realized on this game tree, thereby enhancing its capabilities.

Conclusion

We have demonstrated the ability to adapt a static game tree to the highly dynamic, asynchronous environment typically found with combat fighter aircraft computer generated forces (CGFs). The positive results of our initial tests showed that such an asynchronous game tree could in fact perform well in such an environment when the maximum processing time per decision cycle is constrained. This asynchronous game tree algorithm can be easily applied to a CGF of any domain (land, sea, air, space) by incorporating specifically-tailored knowledge bases of maneuvers for such a CGF. The existing game tree may be enhanced by increasing the breadth of the knowledge base to include more high aspect, offensive, and defensive maneuvers for both the subject aircraft and its adversaries, and allowing maximization of the adversary's move to effect a true mini-max process.

References

1. Winston, Patric Henry. eds. 1993. *Artificial Intelligence, 3rd Ed.* Reading, MA: Addison-Wesley.
2. Schaper, G.; Pandari, S.; Singh, M.1994. Lookahead Limits of Intelligent Player. In Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation, 401-410. IST
3. Zurita, Capt. V. B. 1996. An Architecture for Computer Generated Forces in Complex Distributed Virtual Environments. MS thesis, School of Engineering, U.S. Air Force Institute of Technology.
4. Benslay, 1st Lt. J. 1996. A Domain Independent Framework for Developing Knowledge Based Computer Generated Forces. MS thesis, School of Engineering, U.S. Air Force Institute of Technology.
5. Santos, E. Jr.; Banks, S. B.; Stytz, M. 1996. Engineering Intelligent Computer Generated Forces, Technical Report, School of Engineering, U.S. Air Force Institute of Technology.
6. Edwards, Capt. M. 1996. The Automated Wingman: A Computer Generated Companion for Users of DIS Compatible Flight Simulators. MS thesis, School of Engineering, U.S. Air Force Institute of Technology.