Accepted for publication at ACC 2005; Contact the author(s) for citing permission.

Implementing and Testing a Nonlinear Model Predictive Tracking Controller for Aerial Pursuit/Evasion Games on a Fixed Wing Aircraft

J. Mikael Eklund, Jonathan Sprinkle, H. Jin Kim and Shankar Sastry

***Abstract*— The capability of Unmanned Aerial Vehicles (UAVs) to perform autonomously has not yet been demonstrated, however this is an important step to enable at least limited autonomy in such aircraft to allow then to operate with temporary loss of remote control, or when confronted with an adversary or obstacles for which remote control is insufficient. Such capabilities have been under development through Software Enabled Control (SEC) program and were recently tested in the Capstone Demonstration of that program. In this paper the final simulation and flight test results are presented for a Non-linear Model Predictive Controller (NMPC) used in evasive maneuvers in three dimensions on a fixed wing UAV for the purposes of pursuit/evasion games with a piloted F-15 aircraft.**

I. Introduction

Unmanned Aerial Vehicle (UAV) have recently been used with great success in gathering military intelligence [1] and they provide a viable alternative to manned aircraft due to their smaller size, reduced risk of loss of life, and smaller expense. These factors have invigorated research into UAV autonomy.

However, UAVs have thusfar been used with very little autonomy, and in the face of adversaries this lack of autonomy is a liability. While some defense again ground based adversaries can be achieved through either low level or level flight, success against an intelligent (i.e. manned) airborne adversary must rely on one four possible dimensions in which to obtain an advantage: speed, maneuverability, munitions, and intelligence of control. This experiment focuses on the last of those by improving the intelligence of the aircraft, which allows for current aircraft designs to be reused with software changes.

Nonlinear model predictive control (NMPC) is promising as a control technique that explicitly addresses nonlinear systems with constraints on operation and performance. Previously the use of NMPC has been shown to be effective for rotary-wing UAVs [2]. Aerial vehicles, with their nonlinear dynamics and input/state constraints to guarantee adherence to safe flight, are a proving ground for this technology. However, the use of these control methods that run in real-time on fixed-wing UAVs has not been shown previously.

This research was supported under DARPA's IXO SEC program, under contract number DARPA SEC F33615-98-C-3614.

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Berkeley, CA 94720, USA {sprinkle,eklund,jin,sastry}@eecs.berkeley.edu

Fig. 1. The Global Hawk Medium Altitude Long Endurance UAV (photo courtesy of US Department of Defense).

In this paper, the results of the final integration and testing for the Capstone Demonstration of the Software Enabled Controls (SEC) program are presented for the fixed wing pursuit/evasion games. A numerically efficient nonlinear model predictive tracking control (NMPTC) algorithm is used to encode the pursuit/evasion game between two fixed-wing adversaries. This follows on earlier work [3] and the approach of [4]. The control problem is formulated as a cost minimization problem in the presence of input and state constraints. The minimization problem is solved with a gradient-descent method, which is computationally light and fast [4]. The NMPTC controller uses an interface to an existing autopilot in order to influence the system behavior. By formulating the cost function to include the state information of the other aircraft, input saturation, state constraints and flight test boundary constraints, we show the performance of the NMPTC as a one-step solution for trajectory planning and control of UAVs competing in a pursuit/evasion game.

This paper details the experimental results of flight tests using an NMPTC controller that was designed to provide evasive maneuvers to a fixed-wing UAV when confronted by an airborne adversary of *a priori* type. Section II gives a brief description of the details of the fixed-wing aircraft used for flight and testing, and the expression of the UAV's dynamic and kinematic description. Section III describes the rules of the pursuit/evasion game, along with strategies for success

by the pursuer/evader. Section IV-A gives a detailed description of the encoding of the pursuit/evasion game requirements into the controller, as well as implementation details for rapid simulation of the controller outside the provided testbed. Section V gives the results of some games using the controller, and Section VI presents our conclusions and continuing work.

II. Development and Test Platforms

A. Aircraft details

A Boeing Aircraft Company owned T-33 (originally manufactured by the Lockheed Martin Company) two-seater jet trainer was modified by Boeing for use in the live flight testing in June 2004, and functioned as a UAV surrogate aircraft. This will be referred to as the UAV throughout this paper. The T-33 included a third party autopilot system which did not include airspeed control of the aircraft. The T-33 had on board a safety pilot who could take control of the aircraft in the event of controller malfunction or poor decision making, and well as for controlling the airspeed of the aircraft based on indicator alerts and a displayed target airspeed given to the pilot to increase/decrease thrust. The route and trajectory of the UAV was controlled by a CORBA-based experimental Technology Developer (TD) applications running on a laptop PC with a Linux operating system and Boeing's Open Control Platform (OCP) that was interfaced to the avionics of the aircraft. The TD applications sent the control commands to the avionics pallet that transformed them into autopilot maneuver commands. The state of the UAV, as well as the state of the other aircraft (an F-15 which exchanged state data with the T-33 on a wireless link), was available via this avionics interface. The details of the avionics interface, the available state information, and the input controls are given in the rest of this subsection.

The NMPTC for pursuit/evasion games described in this paper was one of the TD applications developed and tested for this UAV as part of this SEC Capstone Demonstration program.

B. Software in-the-loop simulation testbed

In order to facilitate development, reliable testing, rapid integration, and a uniform interface independent of operating system, a software in-the-loop simulation (SILS) platform was provided by Boeing. This interface used Boeing's Open Control Platform (COP), a generic (black-box) aircraft simulator called DemoSim and Java based UAV Experiment Controller (UEC) GUI. The final versions of OCP and the UEC used were identical to the ones that would be used in the final flight test experiments. This interface provides state information based on the DemoSim model of the UAV, as well as the F-15, to the NPMTC controller and the various other experimental applications that uses it. In addition, a high-level interface to the simulated UAV autopilot is provided that allows the interfacing application to control the rate of change of heading, altitude, and velocity. This included a model of the T-33 test pilot's implementation of the airspeed commands from the NMPTC controller, the functionality missing from the third-party autopilot described above. This is just one example of model-mismatch for which the NMPTC controller would have to demonstrate sufficient robustness.

The complex dynamics of the UAV are not captured in precise mathematical form, due to the imprecise knowledge of the autopilot's behavior with respect to the effect of input on state. In order to provide interfacing applications with a realistic idea of the behavior of the UAV with regard to certain inputs, Boeing also provided a "black box" simulation interface—known as DemoSim—that responds (in real-time) to autopilot commands and outputs the aircraft state data. Using this DemoSim interface, it is possible to test the behavior of aircraft controllers offline, and still have confidence in the results of those tests.

C. Hardware in-the-loop simulation testbed

All software developed for the test flight was provided to Boeing for integration and testing in their Hardware In-the-Loop Simulator (HILS). The HILS system included an interface to the same avionics system used on the UAV and a proprietary aircraft simulation system. Along with software, the experiment test plans were provided with which Boeing developers tested and verified all the software for the various experiments, including the NMPTC controller for pursuit/evasion.

One of the major limitations on the HILS testbed was again, the lack of a human piloted pursuit aircraft. Additionally, the modification to the DemoSim aircraft simulation that allowed for the pursuer to use a similar NMPTC controller as the evader aircraft could not be integrated into the HILS system and the pursuer aircraft could only execute a preprogrammed, non-responsive search pattern. This meant that only the basic functionality of the NMPTC controller could be verified. The actual performance of the controller in a real pursuit/evasion game could only be evaluated during the actual flight test.

D. Vehicle modeling

As described above, the T-33 was simulated with the DemoSim executable that was provided. This simulator included a high level interface to the control the aircraft through the autopilot, and no model of the aircraft nor the simulator was provided. So, a model was constructed based on the states and inputs available and testing of the DemoSim to determine appropriate parameters for this model.

1) State Vector: The overall system state vector, $\mathbf{x}$, is defined using the following equations.

$$\mathbf{x} = \left[\mathbf{x}^K, \mathbf{x}^D\right] \in \mathbb{R}^{n_x} \tag{1}$$

The vector $\mathbf{x}$, which is the overall system dynamics, is partitioned in (1) into the kinematics (denoted by the superscript K) and system-specific dynamics (denoted by the superscript D) matrices. The kinematics of the system is given as the current state of the system in 3 dimensional space, and with respect to the 3-axis posture of the body.

$$\mathbf{x}^K = [x, y, z, \phi, \theta, \psi] \tag{2}$$

The kinematics is shown in (2), where (x, y, z) is the position of the center of mass in 3 dimensions, ϕ is the roll, θ is the pitch, and ψ is the yaw. The dynamics of the system is given as the time rate of change of the kinematic state variables, along with incidental changes, which are represented in classical notation as

$$\mathbf{x}^D = [u, v, w, p, q, r]\,, \tag{3}$$

where $u = \dot{x},\ v = \dot{y},\ w = \dot{z},\ p = \dot{\phi},\ q = \dot{\theta},\ r = \dot{\psi}$. Two state variables, the angle of attack and the angle of sideslip, are absent due to the lack of sensors available on the aircraft, and the autopilot's ability to guarantee heading and attitude of the aircraft.

2) Input vector: The input state vector, $\mathbf{u}$, which is the space of possible inputs to the controller to modify the system state, is determined by the autopilot interface through which we have control of the system (as previously described). We define the input state vector as,

$$\mathbf{u} = \left[u_{\dot{v}}, u_{\dot{\psi}}, u_{\dot{z}}\right] \in [-1, 1]^3 \in \mathbb{R}^{n_u} \tag{4}$$

where $u_{\dot{v}}$ is the desired rate of change of airspeed velocity, $u_{\dot{\psi}}$ is the desired rate of change of turn, and $u_{\dot{z}}$ is the desired rate of change altitude. The input space is constrained by the $[-1, 1]^3$ matrix. However, the actual values sent to the input controller are linearly scaled from the range $[-1, 1]$, such that the maximum outputs to the autopilot were as follows: 50ft/s for airspeed, $\pi/50$ rad/s for turn rate and 10 ft/s for rate of climb to match the approximate limits of the DemoSim.

In addition, boundaries for the values of the state vector, $\mathbf{x}$, are integrated into the optimization cost function to prevent flight out of the test range and autopilot flight envelope, and to prevent violation of the minimum or maximum safe values for speed and altitude.

III. The Pursuit/Evasion Game

In the basic pursuit/evasion game (where the UAV plays the part of the *evader*, and the F-15 plays the part of the *pursuer*), there are asymmetric objectives for the pursuer and evader. The objective of the evader will be to either,

- fly for a predetermined period of time, T, since the start of the game; or
- exit the test range at an opposite corner without being targeted by the pursuer.

The objective of the pursuer will be to,

- target the evader before the end of the game.

The reason for the time limit, T (20 minutes, for the purposes of this paper), is to prevent a trivial solution by the pursuer of haunting a position near the exit point to target the evader on exit. In our game, the pursuer can target the evader by aligning its heading with that of the evader and locating itself within a spherical cone (of predefined height, angle, and diameter) aligned with the tail of the evader. This cone, and a similar one for the pursuer, can be described by the angle off the tail (AOT) and angle off the nose (AON). The AOT is defined by

$$\text{AOT} = \cos^{-1} \frac{A \bullet B}{|A|\,|B|} \tag{5}$$

where, A is the directional vector of the pursuer's motion, and B is the directional vector of the relative position of the evader with respect to the pursuer. AON is defined similarly with respect to the evader. AOT = 0 corresponds to the pursuer being directly behind the evader, and AON = 0 corresponds to the evader being directly in front of the pursuer (the direction of flight of the former in each case is not considered).

The pursuit/evasion game [5], [6], [7] is an interesting application of NMPTC. As discussed in [8] there are four major types of strategies (or controls) for the pursuit/evasion game—open loop, state feedback, nonanticipative, and anticipative. In this experiment the *open loop* strategy is utilized in which the players decide their input signals without any knowledge of the other player's input vector. The other strategies assume progressively more knowledge of the other players input vector (i.e. intentions). A *state feedback* strategy allows knowledge of each player's *current* input vector during the decision process. *Nonanticipative* strategies allow a player to use any input vector of the other player *as long as* the input vector does not correspond to a timestep in the future (with respect to the input being decided, not necessarily the current timestep). Finally, *anticipative* strategies place no restrictions on the current or future values of the input vectors being analyzed. Of course, it is important to note that the input vectors under consideration are only *predicted* inputs, and that they are subject to change if the other players use any of the four types of strategies.

IV. Controller Design

A. Controller Design for the Evader

NMPC problems, in general, consist of the following steps: (1) solve for the optimal control law starting from

the state $\mathbf{x}(k)$ at time k; (2) implement the optimal input $\mathbf{u}(k), \cdots, \mathbf{u}(k+\tau-1)$ for $1 \leq \tau \leq N$; and (3) repeat these two steps at time $k+\tau$.

The solution for the optimal control law can be found by formulating a cost function and considering it when performing the optimization. As described in [9], [2] it is possible to compose this cost function by using the specific details of the application, and the designers best knowledge of optimal performance of the object being tracked. Computational speed, and method, of this technique is discussed in detail in [2]. For our design, we chose the timestep $\tau = 1[s]$, and a lookahead length of $N = 30$ steps. This was a threefold increase from the initial value reported in [3] as it was found that a look ahead of only 10[s] was insufficient for successfully evading the pursuer. The 30[s] lookahead proved sufficient in simulation to produce good evasion and pursuit tactics in the aircraft. Note, that the 1[s] timestep is the trajectory planning timestep which is used to send commands to the avionics system which does the low level control at a 0.01[s] timestep.

In our application we choose to use an open-loop strategy [8] for the pursuit/evasion game, since we have only state vector (and not input vector) knowledge of the other aircraft, which is provided through direct communication between the two aircraft in this experiment. We therefore encode the controller to use this strategy as a weighted member of its cost function. Taking into account the rules of the pursuit/evasion game we were able to design the evader controller by incorporating our desired outcome of the game and encoding some basic aerial tactics as described in [10]. The desired trajectory of the evader, the location and orientation of the pursuer, the input constraints, and the state constraints, are each a part of the cost function. We set this cost function, J, to be

$$J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{u}, \mathbf{d}), \tag{6}$$

where,

$$\phi(\tilde{\mathbf{y}}_N) \triangleq \frac{1}{2}\left(\tilde{\mathbf{y}}_N^{\mathrm{T}} P_0 \tilde{\mathbf{y}}_N\right), \tag{7}$$

and,

$$\begin{aligned} L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k, \mathbf{d}_k) \triangleq \\ \frac{1}{2}\tilde{\mathbf{y}}_k^{\mathrm{T}} Q \tilde{\mathbf{y}}_k + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}} S \mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^{\mathrm{T}} R \mathbf{u}_k \\ \frac{1}{2}\mathbf{p_1}_k^{\mathrm{T}} B_1 \mathbf{p_1}_k + \frac{1}{2}\mathbf{p_2}_k^{\mathrm{T}} B_2 \mathbf{p_2}_k \\ + \frac{1}{\left(\mathbf{d}_k^{\mathrm{T}} G \mathbf{d}_k\right)^{\frac{1}{2n}}} + \frac{1}{\left(\mathbf{a}_k^{\mathrm{T}} H \mathbf{s}_k\right)^{\frac{1}{2n}}} \end{aligned} \tag{8}$$

In these equations, $\mathbf{x}$ is the state vector, and $\mathbf{u}$ is the input vector. The vector $\tilde{\mathbf{y}}$ is the encoding of the error on the current trajectory, and is defined as $\tilde{\mathbf{y}} \triangleq \mathbf{y}_d - \mathbf{y}$, where $\mathbf{y} = C\mathbf{x} \in \mathbb{R}^{n_{\mathbf{y}}}$. The vector $\mathbf{y}_d$ is the desired trajectory of the aircraft at the given timestep. The vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are the horizontal proximity of the aircraft to the 2 nearest boundaries of the playing area/experimental test range. The vector $\mathbf{d}$ is the proximity danger vector between the evader and its adversary, and $\mathbf{a}$ is the AOT of the pursuer with respect to the evader.

The Q, S, R, B_1, B_2, G, and H square matrices each serve as a balancing factor in the cost function. By modifying their relative values of each of these matrices, it is possible to give more "weight" to certain portions of the cost function. We chose to give values to these matrices so that in an equilibrium condition no single factor would outweigh the others, and the aircraft would continue at the same speed on its heading. For this controller *ad hoc* methods are used to find these weighting factors, which required the ability to perform simulations rapidly to reduce the time of development.

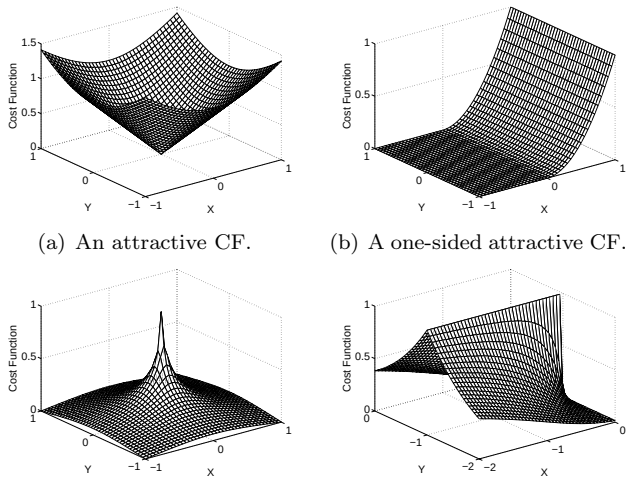
In the definition of $\tilde{\mathbf{y}}$, C acts as a filter to remove elements in $\mathbf{x}$ that are unimportant to the rules of the game. The values for Q differ from those in S, necessarily, as the S matrix is used to ensure that the statically defined constraints on the state vector (e.g., maximum/minimum velocity) are not violated, while the Q matrix is used to ensure that the state values important to winning the game are properly weighted. Furthermore, deadbands are applied to the $\mathbf{x}$ and $\mathbf{u}$ vectors such that S and R only have an affect when one or more states/inputs approach their limits, such as a turn rate limit imposed by the autopilot, minimum and maximum altitude limits imposed (in this case) by the experiment scenario and safety conditions, etc.

The $\mathbf{d}$ vector is the difference in position and heading of the evader and the pursuer. It is used to calculate the proximity of the adversary, and figures into the cost function to outweigh the desired trajectory, should the adversary invade the safe region surrounding the evader. Note that $\mathbf{d}$ contains position information, as well as the angle off tail measurement, which describes the relative relationship of the position of the adversary (regardless of its heading) to the evader's tail. The G matrix, then, is used to appropriately weight this component of the cost function.

The choice of which boundaries to use for the computation of vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ is made by computing the distance from all boundaries (5 in these experiments, as shown in Fig.) and choosing the two nearest. This method is clearly effective only inside a non-convex area. Furthermore, the B_1 and B_2 matrices are set to zero when the aircraft is further than a preset distance from the respective closest range boundary.

Similarly, when the pursuer is more that a preset distance from the evader, the matrix H is set to zero and has no affect on the NMPTC controller.

The first three terms in (8) are quadratic and zero-



(a) An attractive CF. (b) A one-sided attractive CF.

(c) A pointwise repulsive CF (d) A linearly repulsive CF

Fig. 2. Basic forms of the cost functions used by the NMPTC

mean attractive functions. That is to say that they penalize deviation from the zero valued inputs, as shown in Fig. 2(a). The fourth and fifth functions (of $\mathbf{p}_1$ and $\mathbf{p}_2$) are one sided versions of the previous type, that is to say that they penalize deviation in one direction only, as shown in Fig. 2(b). The last two functions are quadratic and zero-mean repulsive functions, meaning that they penalize inputs that approach zero values. The first of those two is a point repulsive function, the second is repulsive along the axis extending behind the aircraft, as shown in Fig. 2(c) and Fig. 2(d) respectively. The forms of these functions are chosen to allow for the easy computation of their derivatives necessitated by the optimization procedure. Note also that the last two (repulsive functions) contain the additional parameter in the order of the denominator polynomial function which affects the shape of the repulsive function.

The solution to the cost-function optimization (using the iterative technique described in [4], [11]) requires the calculation of the derivatives of the vehicle dynamics with respect to both the state and input vectors. Since mathematical equations were not available for the vehicle dynamics—only the DemoSim interface—we used a simplified model using the Eulerian equations of motion, input latency gains and limits on the states to capture our "projected" values for the state of the evader (and pursuer) in the predictive component of the controller.

In order to reduce the computational burden of the nonlinear gradient-descent optimization, the result from the previous time step is used to initialize the optimizer [11], and the a limit on the number iterations is imposed. The second measure may result in a sub-optimal solution being found, however this is generally during a period of rapid change during which (particularly in a pursuit-evasion game) a rapid, sub-optimal decision is preferable to an optimal solution based on estimates of future states made inaccurate by the changing and unpredictable actions of the other aircraft.

B. Controller Design for the Pursuer

An NMPTC controller design for the pursuer aircraft was partially necessitated by the need for an opponent for the evader in the SILS testbed, and also to enable the UAV to assume the role of pursuer.

When in pursuer mode, the UAV used the same controller as in evader mode, however with different matrix values for Q, S and R, the other 5 terms in (8) omitted and the computation for $\tilde{\mathbf{y}}$ was modified for pursuit. Additionally, the pursuers $\mathbf{y_d}$ used to compute $\tilde{\mathbf{y}}$ depended on the pursuers position and relative direction compared to the evader, specifically to offset it's approach prior to turning into a position on the tail of the evader, as discussed in [10]

V. Experimental Results

A. Implementation

The NMPTC algorithm and the pursuit/evasion game were implemented in C++ and run in a Windows environment. The evader aircraft's NMPTC controller was tuned by building it up, component by component of (8) to provide for a successful game outcome in terms of exiting the playing area and avoiding the pursuer aircraft. The trajectory, state and input matrices Q, S, and R were first tuned to ensure that aircraft would follow a set trajectory as both evader and pursuer. Then the evader matrices G and H were tuned for the evader to prevent the pursuer from taking up a targeting position as per the rules of the pursuit/evasion game. Then the boundary matrices B_1 and B_2 along with The former criterion was encoded in the algorithm through the Q matrix of (8), the latter in the G matrix along with the choice of states that were included in the vector $\mathbf{d}$, as described above.

The pursuer algorithm was tuned to close with the evader using its Q matrix and the choice of $\mathbf{y}_d$ as a path toward the predicted position of the evader. Both aircraft controllers used the S and R matrices to constrain the states and inputs.

In the SILS experiments, two instances of the aircraft were simulated, one as the pursuer and one as the evader. For these examples, the performance characteristics of the two aircraft were identical, as was the case in all the simulations leading up to the final test flight due to the constraints of the DemoSim interface.

In the HILS experiments and in the flight test experiments only one aircraft used the NMPTC algorithm. In the HILS a dummy aircraft was used as the other other aircraft to verify the software for flight testing. In the flight test the other aircraft was an F-15 flown by a USAF pilot.

B. Simulation Results

The experiment controller used in the simulation and flight test experiments is shown in Fig. 3 at the start of a SILS experiment. In this case the evader has entered

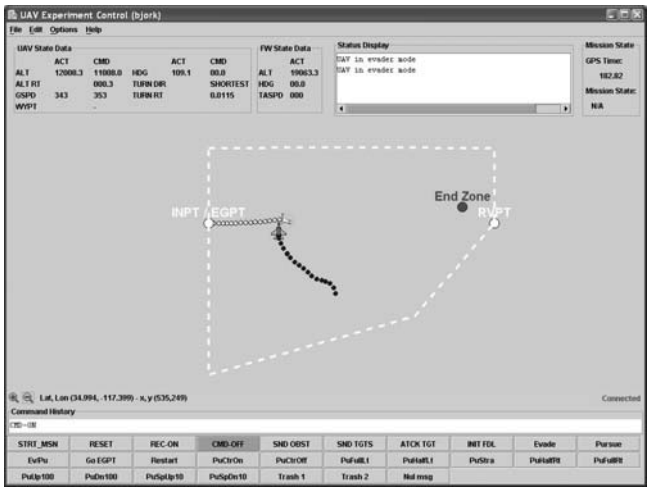


Fig. 3. The UAV Experiment Controller: UAV is yellow, F-15 is blue

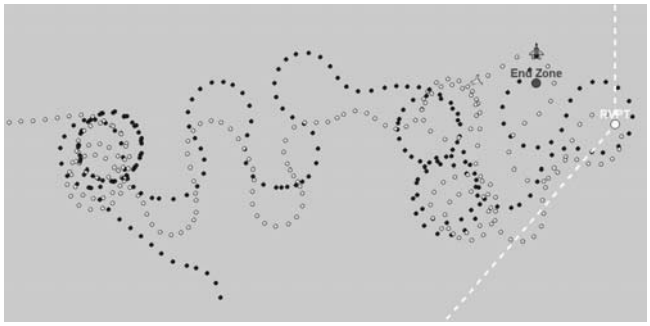


Fig. 4. Detail, a simulation game

from the west and it trying to reach the 'END ZONE' to win the game. The pursuer was in the southern half of the area, turned to engage the evader and then turned to get behind it.

Fig. **??** shows the outcome of this experiment after about 30 minutes of real time. The tracks of the two aircraft can be seen as the evader carries out a series of maneuvers to prevent the F-15 from adopting a targeting position behind it. It can also be seen that the UAV tries to break contact and head for the 'END ZONE' when it feels safe to do so, although it only gradually makes its way in that direction. The effect of the game boundary constraints can also be seen as the UAV's evasive maneuvers bring it close to the southeast boundary. The UAV is forced back into the game area at these times.

In this game, the pursuer win condition was defined by the pursuer being in the 10° AOT cone of the evader (i.e. within 10° of directly behind the evader) while *at the same time* having the evader within the 10° angle off nose (AON) cone of itself (i.e. having the evader within 10° of directly in front). Fig. **??** shows that the evader has successfully avoided losing the game to this point by staying out of these 10° AOT/AON conditions.

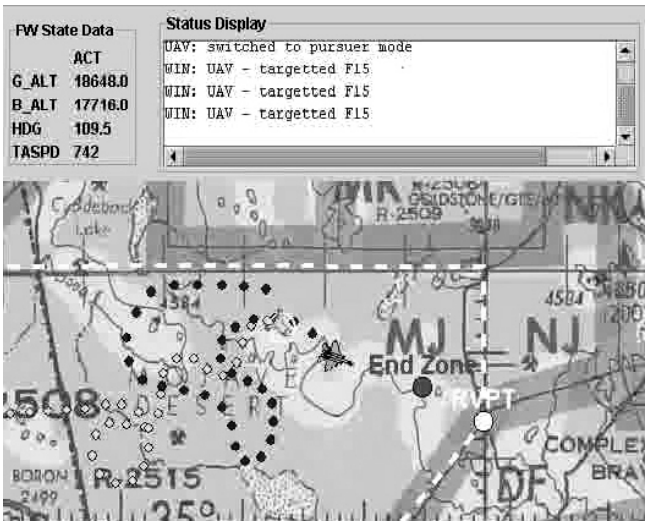


Fig. 5. Flight test experimental results

C. Flight test validation

The NMPTC controller was provided to Boeing for porting to a Linux system for final integration, hardware testing in the HILS and experimental flight tests. These flight tests were carried out as part of the SEC Capstone Demonstration at Edward's AFB during the weeks of June 14-25, 2004. This experiment was run 4 times as part of three different sorties, in which the NPMTC controller and the Boeing platform performed as expected from the simulations, despite significant wind and other conditions.

The first two test flights were flown without the F-15 in order to verify the system. In this case, the same method was used as on the HILS in which the F-15 was simulated. In the HILS it was not possible to implement the pursuer NMPTC controller due to system limitations (i.e. the F-15 model in the SILS did not allow for dynamic control of the aircraft, but pre-programmed control. Boeing modified the SILS system to allow dynamic control of the F-15 in order for the NMPTC controller to be used and the pursuit/evasion games to be tested). In these cases a simulated F-15 was preprogrammed to fly a figure-8 pattern through which the UAV successfully flew to its desired destination virtually unimpeded.

In the third experiment, the UAV was set to only evade. In this case the F-15 was able to get behind the UAV, although did not succeed in targeting the UAV.

In the fourth and final experiment, the UAV was able to switch to pursuer mode if it detected favorable conditions to do so. In this experiment, shown in Fig. 5, the paths of the aircraft can be seen. The F-15 was carrying out a search pattern as the UAV approached. The UAV then detected a advantageous condition and switched to pursuer mode and successfully targeted the F-15. The experimentwas allowed to continued after this (not shown) and the F-15 did shake the UAV off its tail

and managed to get behind it as the UAV switched back to evader mode and tried to reach the 'END ZONE'.

The greatest variable, however, in these experiments was the behavior of the F-15 pilot, who commented very favorably on the behavior of the UAV and the NMPTC controller. Paraphrasing the F-15 pilot after one experiment, the UAV did precisely what one is taught to do in flight school for that situation (in which the F-15 got behind the UAV and tried to adopt a targeting position).

VI. Conclusions and Future Works

A. Conclusions

In this paper we have presented experimental results of the effectiveness of the NMPTC approach to the pursuit/evasion game for fixed-wing aircraft in a time-critical application.

By using the NMPTC approach, rapid computations can be performed, and (given accurate dynamics) the true advantages of autonomy can be encoded using the concepts of competitive games. By providing this autonomous mode (e.g., evader) to a UAV operator, it is possible for a remote operator to relinquish control of the vehicle in time-critical situations, to allow the intelligent controller to serve as a surrogate that incorporates the same theories and behaviors of the pilot. Because the safety and functionality constraints of the aircraft are encoded into the cost function, the UAV is not endangering itself or its environment.

The simulation results show that the encoding of the game into the cost function was successful and these results were validated in actual flight tests on a T-33 UAV surrogate in pursuit/evasion games with a piloted F-15. NMPTC had not yet been demonstrated on fixed-wing aircraft for the pursuit/evasion problem, and this work shows that this method is appropriate when providing input to an autopilot interface.

B. Future Works

The application of NMPTC will continue as we encode the notion of a symmetric pursuit/evasion game into the cost function. This will enable an aircraft to switch roles in the game, thus switching its goals (and associated costs) at runtime. The encoding of this decision to switch roles as part of the cost-function is an exciting possibility. Also, the use of other strategies (besides the open-loop) will be employed for the live demo, to use a variant of the anticipative strategy where a default strategy of the pursuer is assumed by the evader, despite the absence of the actual input vector values.

Currently the game can only be played with the high-overhead avionics interface to the T-33 jet, provided by Boeing. The execution of the code, however, is extremely fast, and future work will involve the hardware implementation in avionics interface to low-cost fixed-wing UAVs.

Proven tactics for evasion and pursuit of fighter aircraft, as discussed in [10], would be useful if encoded into the cost function to encourage these behaviors. Future work into iteratively deriving the weights and values of the cost function matrices could be employed to provide this emergent behavior. Both works are slated for the future, along with simulations that show such behavior emerges from the controller definition.

Finally, the overhead associated with creating a new NMPTC controller is substantial and it would be useful to have a high-level understanding of the controller, as well as a way to generate the controller from this higher level. Providing an abstraction for this level of detail is a useful future research area.

VII. ACKNOWLEDGMENTS

This research funded with the support of the DARPA Software Enabled Control (SEC) program, under contract number DARPA SEC F33615-98-C-3614.

References

[1] AFMC Public Affairs, "Global Hawk UAV supports OEF recon," AFMC News Service Release 1217, December 2002.

[2] H. J. Kim, D. H. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference*, May 2002.

[3] J. Sprinkle, J. M. Eklund, H. J. Kim, and S. Sastry, "Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller," in *Conference on Decision and Control*, December 2004.

[4] G. J. Sutton and R. R. Bitmead, *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory. Basel-Boston-Berlin: Birkhäuser Verlag, 2000, vol. 26, ch. Computational Implementation of Nonlinear Predictive Control on a Submarine, pp. 461–471.

[5] T. Başar and G. J. Olsder, *Dynamic Non-cooperative Game Theory*, 2nd ed. Academic Press, 1995.

[6] M. Bardi, T. Parthasarathy, and T. E. S. Raghavan, Eds., *Stochastic and Differential Games: Theory and Numerical Methods*, ser. Annals of International Society of Dynamic Games. Birkhäuser, 1999, vol. 4.

[7] R. Isaacs, *Differential Games*. John Wiley, 1967.

[8] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Stanford University, Aug. 2002.

[9] F. Allgöwer and A. Zheng, Eds., *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory. Basel-Boston-Berlin: Birkhäuser Verlag, 2000, vol. 26.

[10] R. L. Shaw, *Fighter Combat: Tactics and Maneuvering*. Annapolis, Maryland: United States Naval Inst., 1985.

[11] H. J. Kim, D. H. Shim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots in dynamic environments," Dec. 2003.