43rd IEEE Conference on Decision and Control
December 14-17, 2004
Atlantis, Paradise Island, Bahamas

Encoding Aerial Pursuit/Evasion Games with Fixed Wing Aircraft into a Nonlinear Model Predictive Tracking Controller

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

Abstract— **Unmanned Aerial Vehicles (UAVs) have shown themselves to be highly capable in intelligence gathering, as well as a possible future deployment platform for munitions. Currently UAVs are supervised or piloted remotely, meaning that their behavior is not autonomous throughout the flight. For uncontested missions this is a viable method; however, if confronted by an adversary, UAVs may be required to execute maneuvers faster than a remote pilot could perform them in order to evade being targeted. In this paper we give a description of a non-linear model predictive controller in which evasive maneuvers in three dimensions are encoded for a fixed wing UAV for the purposes of this pursuit/evasion game.**

I. Introduction

The recent success of the Unmanned Aerial Vehicle (UAV) in gathering military intelligence [1] has invigorated research into UAV autonomy. Several reasons that a UAV is a viable alternative to a manned aircraft are its smaller size, reduced risk of loss of life, and smaller expense.

However, UAVs have thusfar been shown to be reliable and effective only against adversaries on the ground (e.g., avoiding small arms fire by flying at high altitudes). In order to be successful against an airborne adversary (either manned or unmanned) there are four possible dimensions in which to obtain an advantage: speed, maneuverability, munitions, and intelligence of control. Increasing the capability of the UAV in any of the first three categories will require either a redesign of the aircraft to increase its payload, maneuverability, engine size, or perhaps all three. Thus, the advantages of a UAV over manned aircraft (size, cost, etc.) are not as stark. However, by improving the intelligence of the aircraft, current aircraft designs may 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. Aerial vehicles, with their nonlinear dynamics and input/state constraints to guarantee adherence to safe flight, are a proving ground for this technology. In fact, in [2] the use of NMPC has been shown to be effective for rotary-wing UAVs. However, the use of these control methods that run in real-time on fixed-wing UAVs has not yet been shown. The need for 'fast' control algorithms for many dynamic systems has previously constricted the implementation of NMPC, since the algorithms must operate in real time.

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

In this paper, following the approach of [3], a numerically efficient nonlinear model predictive tracking control (NMPTC) algorithm is used to encode the pursuit/evasion game between two fixed-wing adversaries. 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 [3]. 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, and state 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 a NMPTC controller that is 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 gives a detailed description of the encoding of the pursuit/evasion game

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

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. Vehicle Model

A. Aircraft details

Due to restrictions on classified data and availability of aircraft, a Boeing T-33 two-place jet trainer was used in the actual flight testing in June 2004, and functioned as a UAV surrogate aircraft (called the UAV throughout this paper, although human test pilots were be aboard for safety reasons). The route and trajectory of the UAV was controlled by an application running on a laptop PC that was interfaced to the avionics of the aircraft. The controller 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.

1) UAV testbed: In order to facilitate reliable testing, rapid integration, and a uniform interface independent of operating system, a CORBA-based platform was provided by Boeing to interface with the avionics of the UAV. This interface provides state information of the UAV, as well as the F-15, to an application that uses it. In addition, a high-level interface to the UAV autopilot is given that allows the interfacing application to control the rate of change of heading, rate of change of altitude, and (through interaction with the pilot) rate of change of velocity. Velocity changes by a controller are not available directly through the UAV auto-pilot, but instead an indicator alerts the pilot to increase/decrease thrust. As such, the timeliness of this input is not dependable during simulation.

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.

2) 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.

3) 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 mapped from the $[-1, 1]$ range to the following ranges,

$$map(u_{\dot{v}}) = \begin{cases} -50[\text{f/s}] & -\infty < u_{\dot{v}} < -1 \\ [-50, 50]\,[\text{f/s}] & -1 \leqslant u_{\dot{v}} \leqslant 1 \\ 50[\text{f/s}] & 1 < u_{\dot{v}} < \infty \end{cases}$$

$$map(u_{\dot{\psi}}) = \begin{cases} -\pi/50[\text{s}^{-1}] & -\infty < u_{\dot{\psi}} < -1 \\ [-\pi/50, \pi/50]\,[\text{s}^{-1}] & -1 \leqslant u_{\dot{\psi}} \leqslant 1 \\ \pi/50[\text{s}^{-1}] & 1 < u_{\dot{\psi}} < \infty \end{cases},$$

$$map(u_{\dot{z}}) = \begin{cases} -10[\text{ft/s}] & -\infty < u_{\dot{z}} < -1 \\ [-10, 10]\,[\text{ft/s}] & -1 \leqslant u_{\dot{z}} \leqslant 1 \\ 10[\text{ft/s}] & 1 < u_{\dot{z}} < \infty \end{cases} \tag{5}$$

where each of the mapped values are in the indicated units. The NMPTC controller will provide values in the $[-1, 1]$ range, allowing the bounds of the mapping to be modified throughout simulation and testing of the controller to produce the desired results.

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

$$\text{AOT} = \cos^{-1} \frac{A \bullet B}{|A|\,|B|} \tag{6}$$

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 [4], [5], [6] is an interesting application of NMPTC. As discussed in [7] 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).

IV. 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 [8], [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 $\tau = 0.333[s]$, and a lookahead length of $N = 30$ steps. This timestep length is based on the clock rate of the interface with DemoSim, which receives the control commands and performs the low level control of the aircraft with a 10[ms] clock rate.

For this paper, we have included the design *only* of the evader, since our online controller will primarily be playing this role. However, we have also created a similar controller for the pursuer, for our own testing purposes.

In our application we choose to use an open-loop strategy [7] 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. 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

$$J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{u}, \mathbf{d}), \tag{7}$$

where,

$$\phi(\tilde{\mathbf{y}}_N) \triangleq \frac{1}{2}\left(\tilde{\mathbf{y}}_N^{\mathrm{T}} P_0 \tilde{\mathbf{y}}_N\right), \tag{8}$$

and,

$$L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k, \mathbf{d}_k) \triangleq \frac{1}{2}\left(\tilde{\mathbf{y}}_k^{\mathrm{T}} Q \tilde{\mathbf{y}}_k + \mathbf{x}_k^{\mathrm{T}} S \mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}} R \mathbf{u}_k + \frac{1}{\left(\mathbf{d}_k^{\mathrm{T}} G \mathbf{d}_k\right)^{\frac{1}{4}}}\right) \tag{9}$$

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 vector $\mathbf{d}$ is the proximity danger vector between the evader and its adversary.

The Q, S, R, G, and P_0 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.

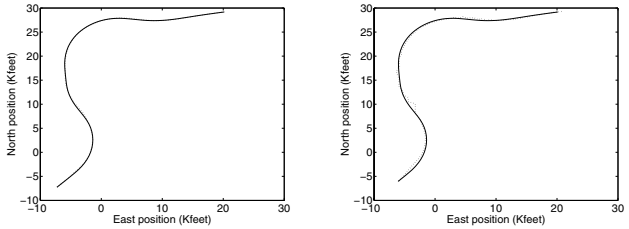
The $\mathbf{d}$ vector is the difference in position and heading of the evader and the pursuer. It is used to calculate the proximity of the adversary, and figures into the cost function to outweigh the desired trajectory, should the adversary invade the safe region surrounding the evader. Note that $\mathbf{d}$ 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 solution to the cost-function optimization (using the iterative technique described in [3], [9]) 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 [9], 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.

A. Simulating in accelerated time

The DemoSim environment—our authoritative model of the aircraft dynamics—provides a state vector response to inputs in real-time. This means that when using DemoSim with our controller, our operations occur at the same rate as they would on the actual aircraft. When configuring the relative weights of the cost-function elements, however, running simulations for 10 minutes at a time meant that we could make fewer than 6 changes per hour to the weights. As such, it was necessary to simulate in an accelerated timeframe to allow for rapid feedback when manipulating the matrix values to balance the cost function.



(a) Look-ahead of 5 steps (b) Look-ahead of 30 steps

Fig. 2. Comparison between the predicted behaviour of the aircraft used by the controller (dashed line), and the DemoSim "actual" behaviour (solid line).

By using our simplified definition of the equations of motion, it was possible for us to approximate the behavior of the aircraft (first in two dimensions, and then in three) using our predictive representation. This allowed us to execute a simulation time of 10 minutes in less than 10 "wall-clock" seconds, thus increasing our changes per hour by a factor of 60.

B. Matching dynamics with DemoSim

It was important, however, to ensure that our simplified dynamics matched the DemoSim dynamics within an acceptable margin of error. Confirmation that our simplified dynamics were reasonably close would enable us to make accurate decisions for matrix weighting during rapid simulation/resimulation phase, and would also show that our predictive model was reasonably close to how the aircraft would actually behave. By modifying the latency gains for the Eulerian equations, we were able to get a reasonably accurate dynamics model. Fig. IV-B shows the aircraft behaviour predicted by our model and used by the NMPTC controller at various look-ahead ranges, versus the DemoSim model behaviour, for a simulation of time 3 minutes.

V. Simulation Results

A. Implementation

The NMPTC algorithm and the pursuit/evasion game were implemented in C++ and run in a Windows environment. 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 being identical, as was the case in all the simulations leading up to the final test flight due to the constraints of the DemoSim interface.

The evader aircraft's NMPTC controller was tuned to provide for a successful game outcome in terms of

exiting the playing area and avoiding the pursuer aircraft. The former criterion was encoded in the algorithm through the Q matrix of (9), 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 aircraft 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.

B. Results

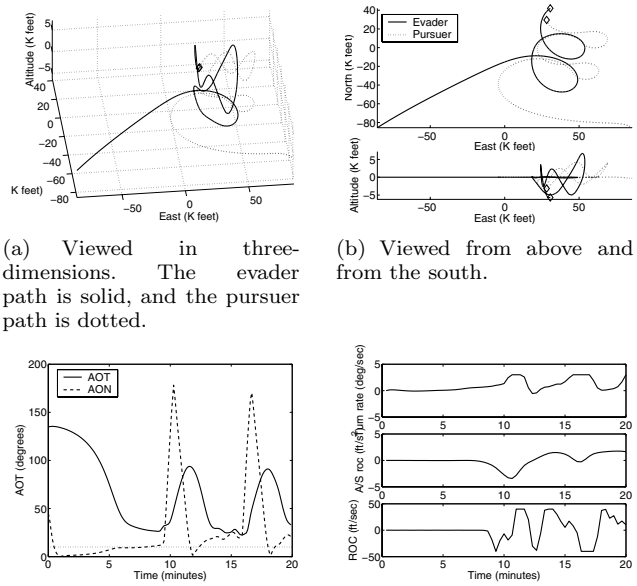
A typical pursuit/evasion game result is shown in Fig. 3(a). In this case the evader has entered in the southwest corner and it trying to exit from the northeast corner of the playing area. The pursuer has entered in the southeast corner and immediately begins to close with evader. In this result, the evader has chosen to turn in the pursuer's direction to avoid allowing the pursuer to take a shooting position on its tail. Turning and climbing/descending maneuvers follow during which the evader is still trying to proceed toward the egress (exit) point to the northeast until the game time expires after 20 minutes. 2-dimensional views are also provided in Fig. 3(b) for better viewing of the aircraft flight paths.

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. 3(c) shows that the evader has successfully avoided losing the game to this point by staying out of these 10° AOT/AON conditions.

The commands generated by the controller are shown in Fig. 3(d). These commands were sent to the autopilot in the DemoSim test platform, and to the T-33 autopilot in the actual test flight.

The behavior of the NMPTC algorithm can be understood by examining individual cost function components that are produced in these maneuvers, and that it is trying to minimize. Fig. 4(a) shows the trajectory component of the cost function ($\tilde{\mathbf{y}}_k^{\mathrm{T}} Q \tilde{\mathbf{y}}_k$, for N steps, varying by the simulation time), in which the function has a small value while it follows its path to the egress point early in the game, then increases and varies as the evader is forced off the trajectory by the pursuer. In this and the following cost function plots, the predicted cost function calculated at each time step can be seen as slices cut across the "Simulation time" axis. These cost functions use a 30 step look-ahead and the simulations are run for 20 minutes simulation time.

Fig. 4(b) shows the cost function of the evader's proximity to the pursuer and Fig. 4(c) shows the separate cost of AOT, which is the cost function that the evader's controller uses to prevent the pursuer from taking a shooting position on its tail.



(a) Viewed in three-dimensions. The evader path is solid, and the pursuer path is dotted.

(b) Viewed from above and from the south.

(c) Win conditions for the pursuit/evasion game (AOT and AON must both be less than 10°).

(d) Autopilot commands generated by the controller, consisting of the turn rate, airspeed rate of change (roc) and rate of climb (ROC).

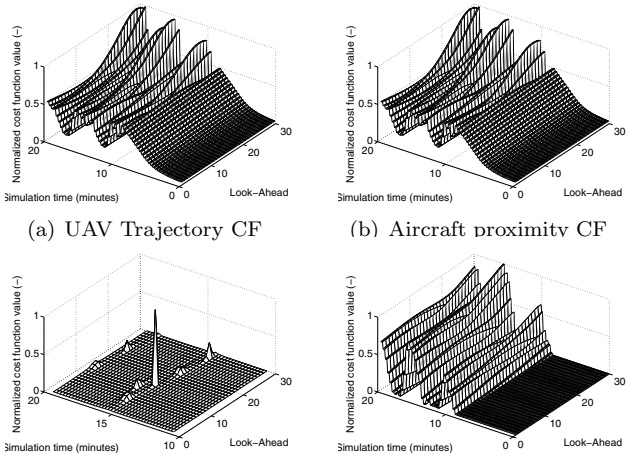
Fig. 3. An example run of the pursuit/evasion game viewed in three-dimensions

The constraints on the states of the evader as shown in the cost function in Fig. 4(d) in which the NMPTC is accounting for limitations in altitude, speed, etc. in its determination of the aircraft's trajectory.

Comparing these figures also allows one to understand how the game developed. For example, the value of the proximity penalty was large slightly before 10 minutes (Fig. 4(b)), and this produces large control commands (Fig. 3(d)), which prevent the pursuer from getting behind it (Fig. 3(c)). This evasion causes the deviation from the desired trajectory (Fig. 3(a) and Fig. 3(b)), which increases the trajectory penalty slightly after 10 minutes (Fig. 4(a)). Also, note how that AOT cost function in Fig. 4(c) increases in the look-ahead around 14 minutes, and how the maneuvers that result successfully prevent this cost from approaching the simulation time axis (i.e. actually happening).

C. Flight test validation

The NMPTC controller was provided to Boeing for porting to a Linux system for final integration, hardware testing 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 greatest variable, however, in these experiments was the behavior of the F-15 pilot, who, although successful in each encounter with the



(a) UAV Trajectory CF (b) Aircraft proximity CF

(c) Angle off tail (AOT) CF (d) UAV State CF

Fig. 4. Cost functions (CFs) for NMPTC

UAV, 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). Details of these experiments are being prepared for publication.

VI. Conclusions and Future Works

A. Conclusions

In this paper we have shown 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] 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.

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

[5] 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.

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

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

[8] 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.

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

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