# Switched and Symmetric Pursuit/Evasion Games Using Online Model Predictive Control With Application to Autonomous Aircraft

J. Mikael Eklund, *Member, IEEE*, Jonathan Sprinkle, *Member, IEEE*, and S. Shankar Sastry, *Fellow, IEEE*

***Abstract*—This paper describes a supervisory controller for pursuit and evasion of two fixed-wing autonomous aircraft. Novel contributions of the work include the real-time use of model-predictive control, specifically nonlinear model predictive tracking control, for predictions of the vehicle under control, as well as predictions for the adversarial aircraft. In addition to this inclusion, the evasive controller is a hybrid system, providing switching criteria to change modes to become a pursuer based on the current and future state of the vehicle under control, and that of the adversarial aircraft. Results of the controller for equally matched platforms in actual flight tests against a US Air Force trained F-15 test pilot are given. Extensive simulation analysis of the symmetric games is provided, including regressive analysis based on initial conditions of height advantage, and relative velocity vectors, and in particular the effect of allowing the evading aircraft to switch modes between "evader" and "pursuer" during the game.**

***Index Terms*—Nonlinear control systems, optimization, path planning, predictive control, unmanned aerial vehicles (UAVs).**

## I. Introduction

RECENT successes of the unmanned aerial vehicle (UAV) in gathering military intelligence [3] have invigorated basic research into autonomy and methods to design intelligent controllers, while preserving safety of operation [4]. UAVs are exciting alternatives to manned aircraft for many purposes, due to its smaller size, reduced risk of loss of life, etc. Further, many UAVs are not fully autonomous, but are rather remote piloted aircraft, which allows for longer mission times as remote pilots can tradeoff without needing to land the vehicle. Efforts to enable autonomous refueling [5]–[8] will further expand mission lengths. Reducing landings also lessens time lost to use-based maintenance and inspection of the aircraft.

UAVs have been shown as reliable and effective against adversaries on the ground, since their high cruising altitude avoids small arms fire. However, if UAVs were to encounter unfriendly aerial adversaries, several issues must be addressed. Such remote vehicles are controlled through sending waypoints, or direct joystick control. With these interfaces, time lag, which can measure on the order of seconds to and from the remote pilot, makes human-controlled maneuvers difficult [9]. It also rules out the application of evasive maneuvers that require tight coupling between sensed, and predicted, locations of the adversary and control inputs, if those inputs are given by a human-in-the-loop. Even if time lag is not a consideration, the lack of situational awareness and visual sensors of the remote pilot means that optimal behavior is difficult, if not impossible, to achieve. Given these conditions, the use of the current controller methods is infeasible for this application, and would lead to quick capture of the UAV.

In order to be successful against an airborne adversary (either manned or unmanned) there are five possible dimensions in which to obtain an advantage: speed, maneuverability, observability, munitions, and intelligence. Increasing the capability of the UAV in any of the first four categories will require either a redesign of the aircraft to increase its payload, maneuverability, form factor, engine size, or perhaps all four. Even discounting the cost of the redesign, changing these parameters reduces the advantages which a UAV has over manned aircraft in size and efficiency. Improving the intelligence of the aircraft allows for current aircraft designs to be reused with software changes and enhancements.

This paper provides a description of such a "best effort" controller, which will improve on the best behavior of a remote pilot due to an autonomous controller's low-latency response when compared to a remote pilot. The paper extends work in [1] and [2] through extended analysis and simulation to evaluate the equilibrium of the game. We begin the paper with a discussion in Section III of the details and background of a pursuit/evasion game. In particular, we describe the roles of the evader and pursuer, along with strategies which each can employ in order to attempt to optimize their performance against the other. This portion of the paper will acquaint the reader with several important conditions and requirements for winning the game, and will

Manuscript received November 30, 2009; revised May 28, 2010; accepted February 14, 2011. Date of publication April 29, 2011; date of current version April 11, 2012. This work was supported under DARPA's IXO SEC Program, under Contract DARPA SEC #F33615-98-C-3614. Funding during the analysis and continuation of the original work was provided by AFOSR Award #FA9550-091-0519, "Modeling of Embedded Human Systems", NSF Award #CNS-0930919, and by the Center for Hybrid and Embedded Software Systems at UC Berkeley, which receives support from the National Science Foundation (NSF Award #CCR-0225610), the State of California Micro Program, and the following companies: Agilent, DGIST, General Motors, Hewlett-Packard, Infineon, Microsoft, and Toyota.

J. M. Eklund is with the Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, ON L1H 7K4, Canada (e-mail: mikael.eklund@uoit.ca).

J. Sprinkle is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721-0104 USA (e-mail: sprinkle@ece.arizona.edu).

S. S. Sastry is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94720 USA (e-mail: sastry@eecs.berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2011.2136435

provide a foundation upon which we can discuss the particular live game demonstration in which we participated with a US Air Force trained test pilot.

Section IV gives a brief review of the important aspects of model predictive control (MPC), particularly those relevant to strategy and the implementation of nonlinear model predictive tracking control (NMPTC) used here. Readers who are unfamiliar with MPC on a technical level, but who perhaps understand it intuitively, will benefit from the mapping we provide between the numerical and computational portions of the controller and its design, which are discussed in Section IV-A, and the definition of the specific game in which we demonstrated the technology. This section applies a methodology which will be useful for future implementations of MPC-based controllers. We also discuss here the dual nature of our controller as both evader and pursuer, and how this enhanced our testing and simulation phases of controller design. We describe the symmetric controller in which the evader can switch modes to become a pursuer, and back again if necessary.

Section V presents the simulation, development and test platforms used in this project, as well as preliminary results used to validate the flight models identified and used. These include software- and hardware-in-the-loop simulations, as well as the flight test platform used.

In Section VI the results of the demonstration are given, and some discussion regarding how the manned vehicle was operated in order to maintain symmetry is provided. We transition into Section VII, where we describe our extensive simulation experiments that were conducted following the flight test validation of the control strategy within the context of these pursuit/evasion games. We also discuss the ramifications of the introduction of this notion of a "switched" game.

## II. Related Work

### A. Constrained Nonlinear MPC

A significant hurdle in the real-time or autonomous application of nonlinear MPC is the guarantee that the control inputs, and predicted paths, will satisfy the constraints of the system. For hard constraints, such as obstacle avoidance or physical limitations of the vehicle, this is a critical effort. Work by Richards and How provides guarantees on completion time of certain maneuvers with the introduction of terminal constraints into the cost function summation [10]. In [11] a similar approach is used to guarantee obstacle avoidance, through the comparison of a basis (or safe) state in which the vehicle must remain throughout the lookahead horizon, and a subsequent guarantee that such basis states do not intersect with any obstacles. A comprehensive treatment of predictive control can be found in [12].

The satisfaction of constraints for these nonlinear dynamics (or perhaps the nonlinear application of constraints) requires sophisticated techniques to produce the optimal control input. Nonlinear programming packages such as NPSOL [13] and TOMLAB [14] are commonly applied in these cases. However, when the constraint set becomes large, finding an optimal value that satisfies such constraints can become computationally infeasible. Work by Chung [15] presents methods that will satisfy such high-constraint (on the order of 640) problems with speedups of up to two orders of magnitude. With such methods, flight in unknown urban environments can be performed with high confidence [16].

In many cases the time required for optimization, or storage required for optimization, becomes the limiting resource for real-time application of nonlinear MPC (linear models and constraints can use dynamic programming to provide control inputs [17]). Recent work by Zeilinger [18] examines how to reduce the approximation in order to produce suboptimal, but bounded-time to compute, control inputs. Related issues of stability are discussed by Morari *et al.* in [19].

### B. Game Theory and Optimal Control

The two-person pursuit/evasion game (PEG) is a differential game with a zero-sum outcome. In [20] this kind of game and several interesting variations are explored and defined, and strategies to solve them are discussed and illustrated. Generally, PEGs [20]–[22] are a well-established abstraction for optimal control.

The concept of a PEG has been previously used as an abstraction for design and/or verification of a safe controller, and was described in [23] as a particular example among several topics in game theory. In such applications, a disturbance is modeled as some pursuer, and the plant is an evader: the job of the controller is to choose a strategy that always lets the evader win (i.e., the plant is in a controllably safe state for all future time). In the game described in this paper, our application does not need this logical layer as our notion of safety is winning or losing of the game itself.

As discussed in [24] there are four major types of strategies (or controls) for the PEG—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.

### C. Aerial Pursuit/Evasion Games

Ménec [25] describes strategies for what are called "Modern Aerial Duels." The PEG in that paper is predicated on medium- or long-range missiles which provide a firing envelope. Traditional strategies for human-piloted aircraft are presented in Shaw's text [26].

Application of real-time games on rotorcraft is discussed in [27]. Various equilibrium points for such games are discussed in [28]. The application of vision as the primary sensor is discussed further in [29].

In this work, our constraints from the physical system (to maintain lift of the fixed-wing aircraft) provides an additional extension of these mission-critical examples, where flight-critical response must be accounted for while simultaneously participating in the game.

### D. Comparison With Existing Approaches

Previous approaches to solving pursuit/evasion games leverage the above results from game theory, fighter tactics, model-predictive control, and optimal control. However, each of these approaches makes some assumption or accommodation that this paper does not relax.

The algorithm presented by Ménec [25] describes a *single*, maneuver in the plane at a particular decision point, but does not provide optimization over some *horizon*. In fact there is only one objective which is to evade targeting by a missile, and there is no constraint which targets arrival at a final location. The work presented in this paper requires a repeated evaluation of whether the evader may be targeted within the prediction horizon, while simultaneously preferring to fly toward a final waypoint.

In [27] the game is also 2-D, though that game provides an interesting perspective of utilizing a rotorcraft to coordinate ground vehicles in the search for an evader. That example has three pursuers searching for the evader, using an estimation policy for determining the evader's location. In that case, the vehicles, like those presented in this work, are considered to be symmetric, but have no constraints on minimum forward velocity-which is crucial for fixed-wing vehicles. In that work, the evader's behavior is random in the plane, and at a speed much slower than the pursuers: thus, capture by the pursuers is expected, given the advantages in number and speed, despite the limited horizon for detection. It is worth noting that the work presented in this paper is in three dimensions, and also has angular constraints for ending the game if one vehicle targets the other.

The work in [28] utilizes a horizon for control of probabilistic games in a plane. However, those games are played using discrete motion across a grid, and do not consider the continuous dynamical constraints of the vehicle. Rather, the standard topological constraint of motion in a grid (moving only to adjacent cells) is chosen, although the speeds of vehicles are used in order to provide advantage to one vehicle over the other. The state of the evader is available to the pursuer, but only in adjacent cells, whereas in the approach of this paper each vehicle has perfect state data from the other.

A search algorithm using fixed horizon based on vehicle parameters is found in [30], and a search strategy similar to a pursuit strategy is based on a probabilistic threat exposure map (PTEM). That work could be used to extend the approach presented in this paper if, in future work, state data were not to be shared by each vehicle. One strength of the work presented in this paper, however, is that evasion (and pursuit) is performed using full state information, and with predictions based on the actual strategies used by the adversary's vehicle; thus, the game's loser cannot claim lack of information as the reason for losing the game, as it is (to some degree) predetermined by initial conditions, and the strategies used by each player.

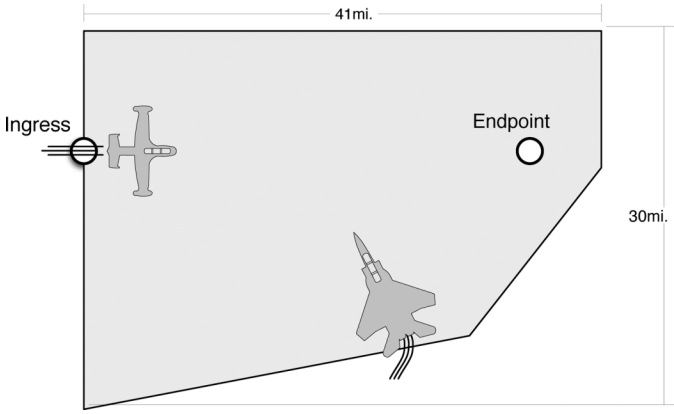


Fig. 1. Game is restricted into some linearly defined space.

One approach closely related to that presented in this paper comes from Shim *et al.* [31], in which MPC is used to control a rotorcraft to track a trajectory, based on an optimal control formalism, while simultaneously avoiding obstacles. In that paper, the constraints on vehicle performance are taken into account; however, a rotorcraft has fundamentally different constraints from a fixed-wing aircraft in terms of minimum velocity and rate of turn. Also, in that work, the optimization algorithm is not performed online.

In summary, the approach presented in this paper differs from previous work that uses a receding-horizon approach in that this paper considers more than one way to win (either elapsed time, or arrival at a final waypoint). Also, this work places constraints on vehicle motion such as a minimum and maximum velocity, and maximum turn rate. Previous work that considers vehicle motion constraints does not consider fixed-wing aircraft constraints, and also does not solve the problem using online optimization. Work that considers analytical solutions to the pursuit/evasion game either does not permit multi-step games, does not consider more than one winning strategy, does not solve the problem in three dimensions, or does not allow a player to switch strategies during game play.

## III. Demonstration Pursuit/Evasion Game Description

All the PEGs discussed in this paper share a common playing area, which is a clearly defined airspace shown in Fig. 1, limited to altitudes between 10 000–20 000 ft. The aircraft are required to remain within this airspace for the duration of the PEG.

In our basic PEG there are asymmetric objectives for the pursuer and evader. The goal of the evader is fulfill one of the following objectives:

- fly for a predetermined period of time, $T$, since the start of the game;
- reach the objective area, called the "Endpoint" or "End Zone" near the opposite corner without being targeted by the pursuer.

The objective of the pursuer is quite simply to target the evader before the end of the game. Both evader and pursuer are further required to remain within the game boundary throughout the game. In the symmetric game, the evader win conditions include the targeting of the pursuer.

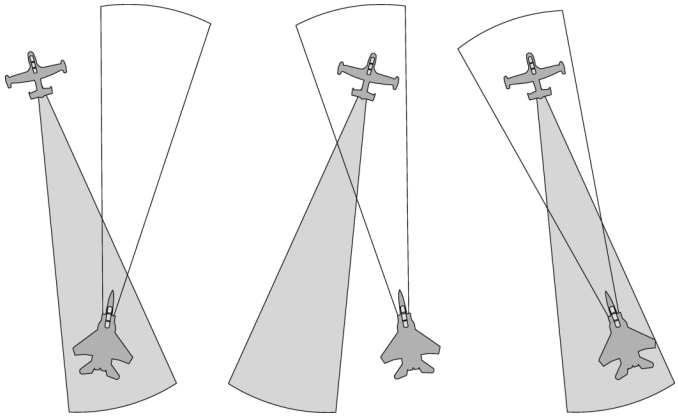

Fig. 2. Targeting rules for these PEGs. In the condition on the left the evader is not in the pursuer's cone. In the middle, the pursuer is not in the evader's cone. On the right, a successful targeting condition is shown with the pursuer behind *and* oriented with the evader.

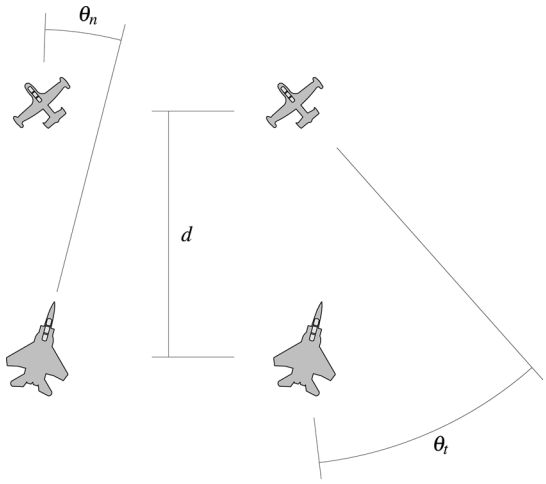


Fig. 3. Angular definitions for the game. $\theta_n$ is the *angle off nose*, defined as the relative heading of the evading aircraft from the pursuer. $\theta_t$ is the *angle off tail*, defined as the relative heading of the pursuing aircraft from the evader. The distance between the centers of the aircraft is $d$.

The time limit $T = 20$ min is imposed to prevent a trivial solution by the pursuer of haunting a position near the exit point to target the evader on exit and to reflect the constraints of the flight test experiments. 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 and angle) aligned with the tail of the evader. This targeting condition is shown in Fig. 2, and in these games a 10° cone of length 3 nm (nautical miles) is used.

We use two angular definitions for the game. The *angle off tail*, $\theta_t$, is the bearing of the pursuer from the evader's tail. The *angle off nose*, $\theta_n$, is the bearing of the evading aircraft from the pursuer's nose. These values can be calculated as

$$\theta_t = \cos^{-1}\left(\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}\right) \tag{1}$$

where $\vec{a}$ is the directional vector of the adversary's (pursuer's) motion, and $\vec{b}$ is the directional vector of the relative position of the evader with respect to the pursuer. $\theta_n$ is defined similarly with respect to the evader (see Fig. 3). $\theta_t = 0$ corresponds to the pursuer being directly behind the evader, and $\theta_n = 0$ corresponds to the evader being directly in front of the pursuer (the direction of flight of the former in each case is not considered except by combining the two angles). Thus, to be "targeted", the vehicles must be in each other's cones, i.e., $\theta_t| < 10°$, $|\theta_n| < 10°$, and $d < 3$ nm.

As will be seen in the results, these objectives and rules for the PEG favor the evader aircraft; however these rules were determined for a flight test demonstration, well in advance of any simulation results. The results and analysis of the flight test and simulation results should be viewed as relative to those validated in the flight test based on various parameters, but not as an attempt to improve the controller design, which is a matter for future work.

## IV. Controller Design

The implementation of the hierarchical MPC-based controller required the development of several components which are described next. These include the vehicle modeling necessary for MPC including the model constraints, and the cost functions for the two modes of operation within the PEGs of interest.

### A. Model Predictive Control

MPC 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 $\mathbf{u}(k), \ldots, \mathbf{u}(k+\tau-1)$ 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 $J$ and considering it when performing the optimization. As described in [31] and [32] 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 [31].

In particular, we follow the approach of [31] and [33] in which MPC is formulated as a nonlinear tracking problem in the presence of input and state constraints. This approach is referred to as NMPTC and provides the trajectory generation layer and tracking control within a hierarchical control scheme. Within the context of this application, this is very well suited as the lower level control is provided by an existing hardware autopilot system which is only accessible through higher level control inputs.

With NMPTC, separate cost functions are used to effect the trajectory generation and tracking control; however both sets of functions are used, as well as the state and input constraint functions, in the same cost minimization step to determine the optimal control decision at each time step. The specific cost functions used in the work and their descriptions are provided in Section IV-D.

The definition of the cost function, $J$, clearly plays a significant role in the behavior of the system for both MPC and

NMPTC. For our purposes in this paper we consider a cost function defined as

$$J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}_k, \mathbf{u}_k, \cdot) \tag{2}$$

where

$$\phi(\tilde{\mathbf{y}}_N) \triangleq \left(\tilde{\mathbf{y}}_N^{\mathrm{T}} P_0 \tilde{\mathbf{y}}_N\right) \tag{3}$$

where $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the input vector. $P_0$ represents constraints on the final state vector $\tilde{\mathbf{y}}$, which 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, and the values for $C$, as well as the remaining inputs to and definition of $L$, are given in Section IV-D following the layout of rules for the pursuit/evasion games.

Optimization of the cost function is performed as $\min_{\mathbf{u}\in\mathbb{U}^{n_u}} J(\cdot)$, which requires a predictor function $x_{k+1} = f(x_k, u_k, \cdot)$, which we later fully define for our application. Using this predictor function, the vector $\mathbf{u}$ (which is of length $N$) provides a control input for $1 \leq k \leq N$. Performing this optimization in real-time is nontrivial, and discussed in Section II.

### B. Vehicle Modeling

Dynamically accurate simulation of the UAV was possible through the DEMOSIM application, provided by Boeing for use in these experiments. As described in more detail in [1], this simulator provided real-time outputs that were guaranteed to match the final physical platform with sufficient accuracy.

*1) State Vector:* The overall system state vector $\mathbf{x}$ is defined as

$$\mathbf{x} = \left[\mathbf{x}^K, \mathbf{x}^D\right] \in \mathbb{R}^{n_x}. \tag{4}$$

The vector $\mathbf{x}$, which is the overall system dynamics, is partitioned in (4) 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-D space, and with respect to the 3-axis posture of the body

$$\mathbf{x}^K = [x, y, z, \phi, \theta, \psi]. \tag{5}$$

The kinematics is shown in (5), where $(x, y, z)$ is the position of the center of mass in three dimensions, $\phi$ is the roll, $\theta$ is the pitch, and $\psi$ 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

$$\mathbf{x}^D = [u, v, w, p, q, r] \tag{6}$$

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

$$\mathbf{u} = \left[u_{\dot{v}}, u_{\dot{\psi}}, u_{\dot{z}}\right] \in [-1, 1]^3 \in \mathbb{R}^{n_u} \tag{7}$$

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 of altitude.

*3) Comparison With* DEMOSIM*:* As previously mentioned, mathematical versions of the physical model were unavailable, and would have presented difficulties in simulation due to complexity. So we selected Eulerian models for the vehicle behaviors, and compared recorded trajectories to determine whether our kinematic models were sufficiently close to the DEMOSIM behaviors.

This 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. 4 shows the aircraft behavior predicted by our model and used by the NMPTC controller at various look-ahead ranges, versus the DEMOSIM model behavior, for a simulation of time 3 min. This simulation time selected chiefly to validate the sufficient approximation of the controller's model to the dynamically accurate DEMOSIM model for various discrete lookahead step sizes of $N$ =5, 10, 20, and 30, each with the discrete timestep $\tau = 1$ s.

### C. States, Constraints, and Measurements

For the live demonstration, the aircraft were to have access to the adversary's state data at each timestep, and this was also enabled within the simulation environment. This was done to keep the issue of sensing and detection separate from the issue at hand, which is that of control and trajectory generation in PEGs. The issue of adversary state uncertainty is a concern for future work. However, the various constraints on the states and inputs of the system were very important to the development and demonstration.

The state space is constrained, for example by minimum and maximum altitudes and airspeeds and the input space is constrained by the existing autopilot capabilities as determined from the DEMOSIM simulator, which were 50 ft/s$^2$ for airspeed rate, $\pi/50$ rad/s for turn rate and 10 ft/s for rate of climb. These input limits were scaled for the NMPTC controller as a $[-1, 1]^3$ matrix.

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.

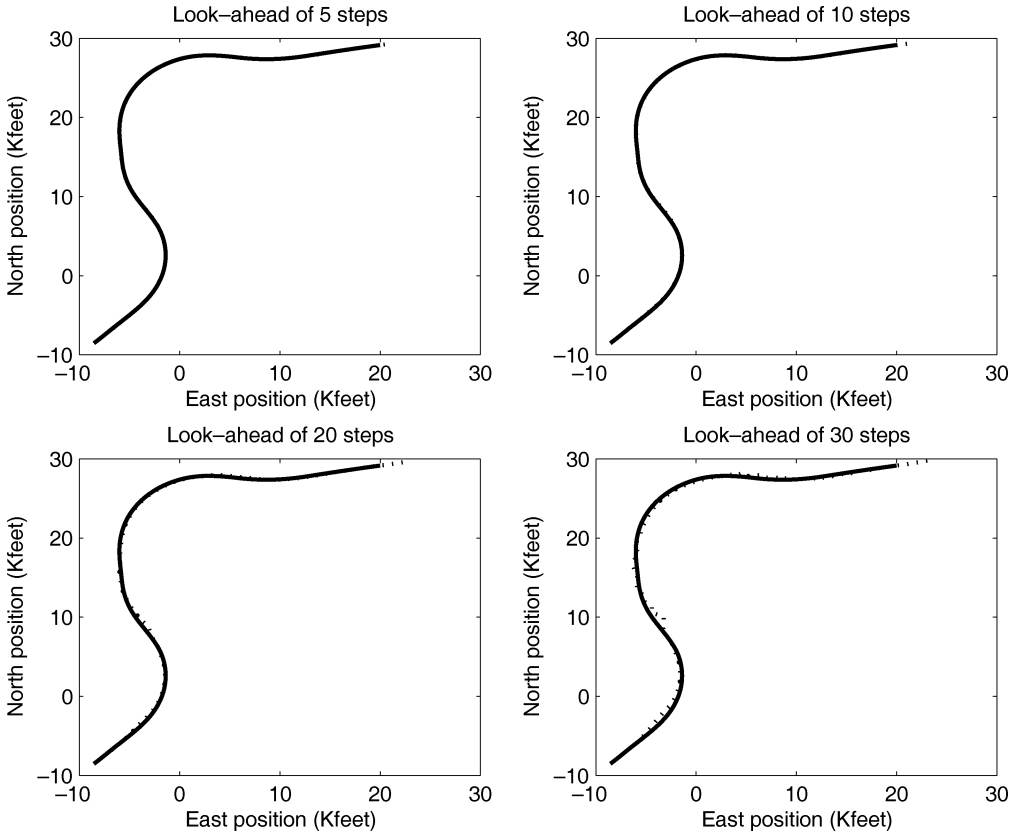


Fig. 4. Comparison between the predicted behavior of the aircraft used by the controller (dashed line), and the DEMOSIM "actual" behavior (solid line).

### *D. Controller Design for the Evader*

Taking into account the rules of the PEG 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 [26]. Additionally, state and game constraints necessary for the flight test were added to the basic controller described in [1]. For our final design, we chose the timestep $\tau = 1$ s, and a lookahead length of $N = 30$ steps. 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.

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 now define values previously abstracted in (2)

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

The vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are the horizontal proximity of the aircraft to the two 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 $\theta_t$ of the pursuer with respect to the evader.

The $Q$, $S$, $R$, $B_1$, $B_2$, $G$, and $H$ square matrices each serve as weighting factors in the cost function. By modifying their relative values, 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 were 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 effect 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

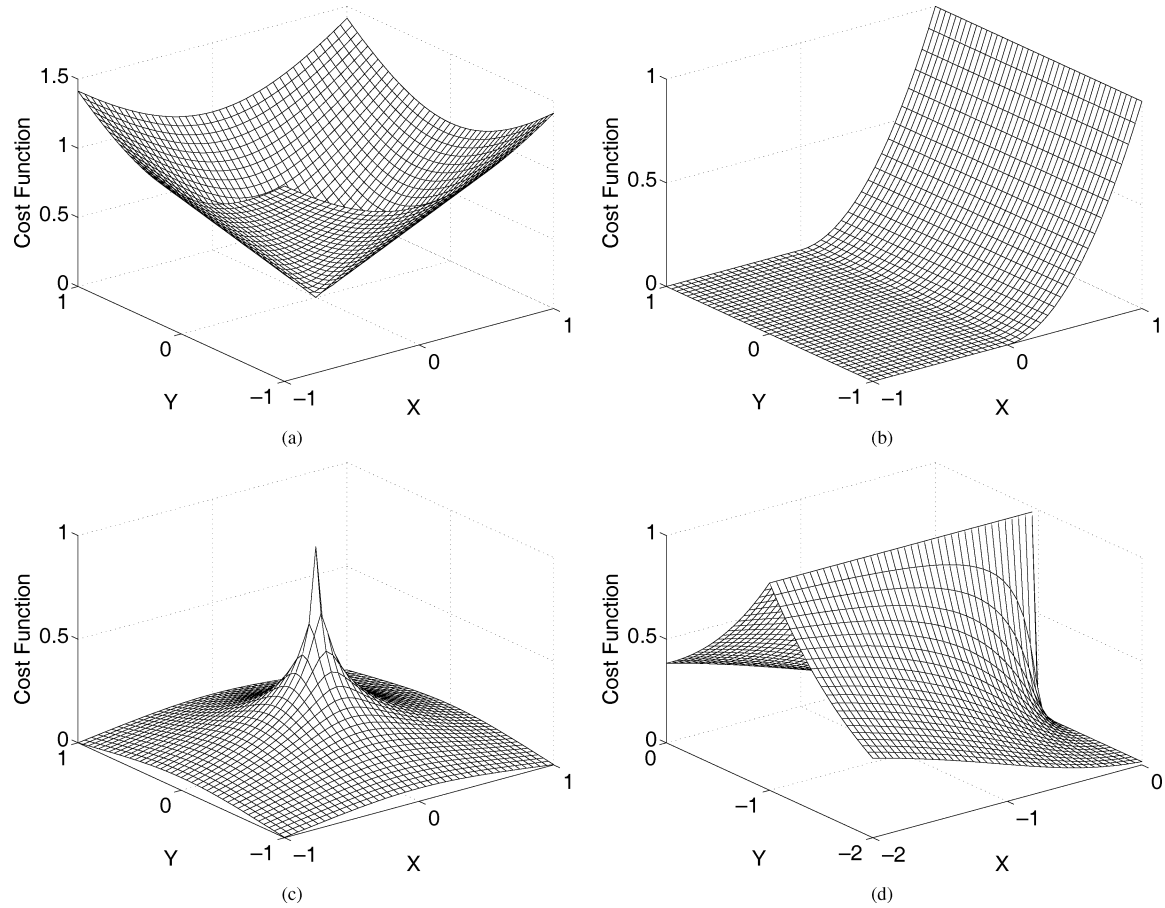


Fig. 5. Basic forms of the cost functions used by the NMPTC. (a) An attractive CF. (b) A one-sided attractive CF. (c) A pointwise repulsive CF. (d) A linearly repulsive CF.

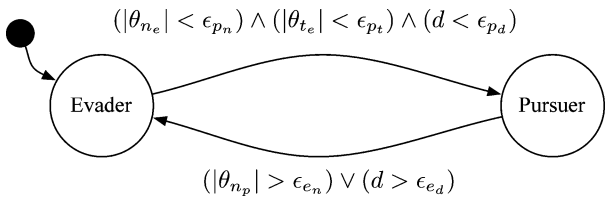


Fig. 6. Conditions for switching between pursuit/evasion parameters.

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 (five in these experiments, as shown in Fig. 11) 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 than a preset distance from the evader, the final cost function element (containing $H$) goes to zero and has no affect on the NMPTC controller.

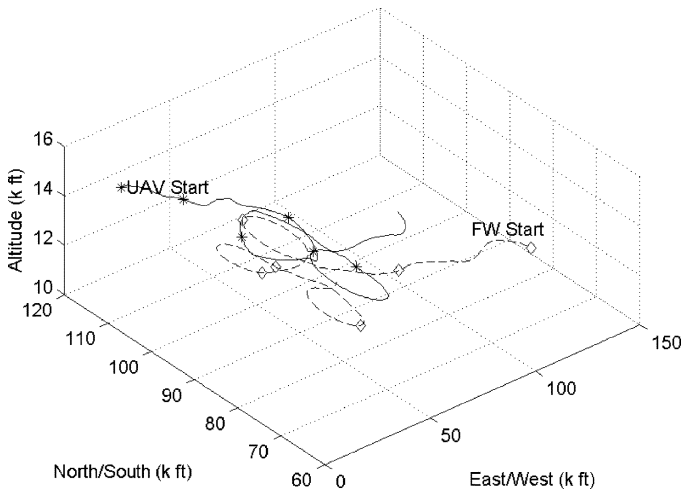


Fig. 7. Pursuit evasion game with the trajectories shown in three dimensions. Only the first 10 min are shown for clarity. Note that the vertical axis is scales about 10 times larger to make it easier to see the flight paths, and roughly approximates the relative forward and vertical speed characteristics of the aircraft.

### *E. Controller Design for the Pursuer*

An NMPTC controller design for the pursuer aircraft was initially necessitated by the need for an opponent for the evader in

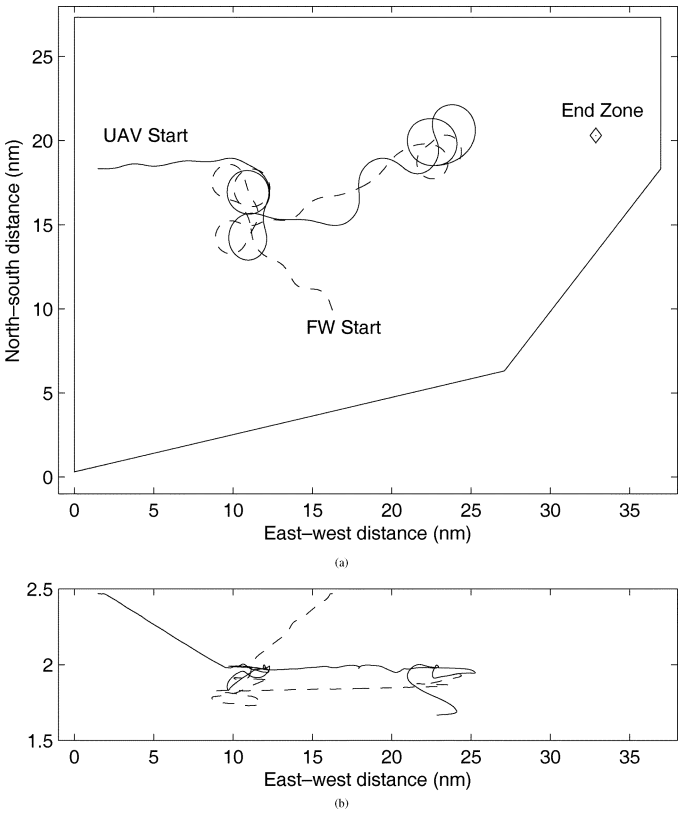


Fig. 8. Same game as Fig. 7 viewed from different perspectives and with the full 20 min shown. Note again the scaling difference of the vertical axis. (a) Top View. (b) Side View (facing North).

the software-in-the-loop system (SILS) testbed, and later to enable the evader UAV to assume the role of pursuer.

When in "pursuer" mode, the UAV used the same controller software as in "evader" mode, but with different matrix values for $Q$, $S$, and $R$, the other terms in (8) omitted and the computation for $\tilde{\mathbf{y}}$ was modified for pursuit. Additionally, the pursuer's $\mathbf{y_d}$ used to compute $\tilde{\mathbf{y}}$ depended on the pursuer's position and relative direction compared to the evader, specifically to offset its approach prior to turning into a position on the tail of the evader, as discussed in [26].

### *F. Topology of Cost Function Elements*

The first three terms in (8) are quadratic and zero-mean attractive functions: they penalize deviation from the zero valued inputs, as shown in Fig. 5(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. 5(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. 5(c) and (d), respectively. The forms of these functions were chosen to allow for the easy computation of their derivatives to facilitate 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.

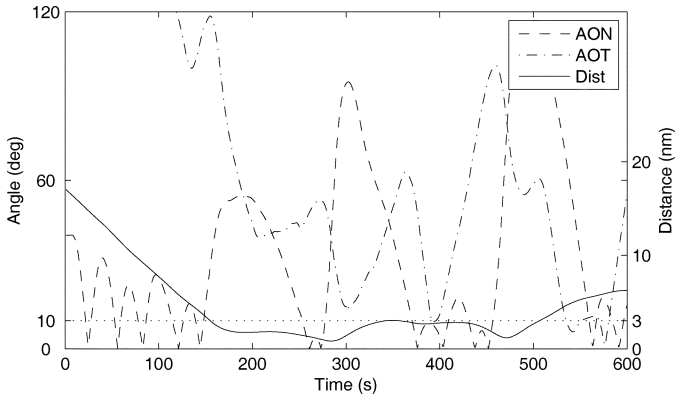


Fig. 9. Win conditions for the pursuit/evasion game ($\mathrm{AOT} = \theta_t$ and $\mathrm{AON} = \theta_n$ must both be less than 10° and the distance less than 3 nm). The first 10 min of the test are shown which included a successful targeting of the evader by the pursuer at 392 s.

The solution to the cost-function optimization (using the iterative technique described in [33] and [34]) 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 [34], and a limit on the number of iterations is imposed. The second measure may result in a sub-optimal solution being found, however this generally occurs during a period of rapid change in which (particularly in a PEG) a rapid, sub-optimal decision is preferable to a delayed optimal solution based on estimates of future states that will most likely be made inaccurate by the changing and unpredictable actions of the other aircraft.

### *G. Switching Between Pursuer/Evader Modes*

In the symmetric version of the PEG, in which the evader is able to target the pursuer to win the game, the aircrafts' distance and $\theta_n$ conditions may trigger a switch to the pursuer NMPTC parameters until such time as the favorable conditions are lost. These parameter bounds are defined as the evader's[1] angle off nose and tail ($\theta_{n_e}$, $\theta_{t_e}$), and the pursuer's same values (shown as $\theta_{n_p}$, $\theta_{t_p}$ for clarity). As shown in Fig. 6, transition occurs from "evader" to "pursuer" mode only if all conditions are satisfied. Transition can return the vehicle to "evader" mode if *any* of the other conditions are satisfied. The conditions can be described, respectively, as evader is in a favorable position to chase the pursuer (opportunity), while at the same time the pursuer is not is a good position to catch the evader (safety) and they are close enough that is it worth trying rather than continuing towards the End Zone objective (benefit).

[1]The $\theta_{n_e}$ is the angle off nose for the inverse of the positions shown in Fig. 3; i.e., the evader is in a more advantageous position than the pursuer.

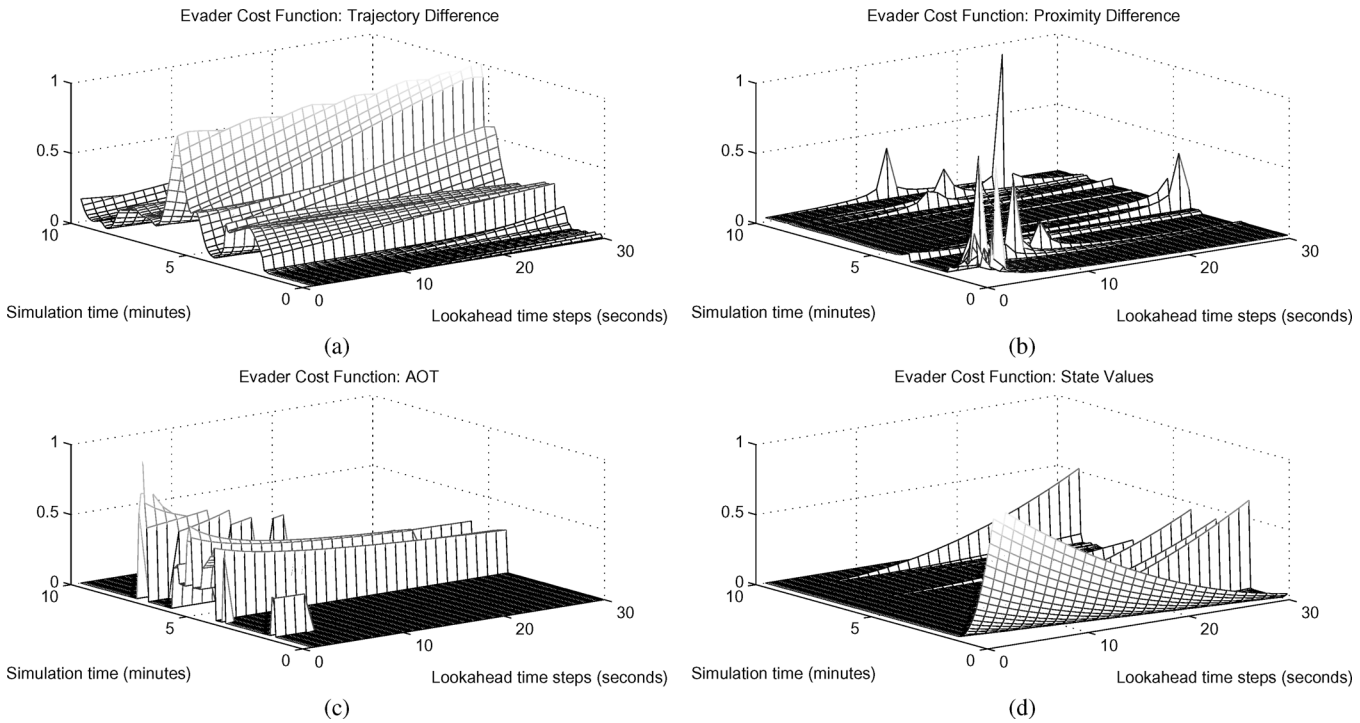


Fig. 10. Experimental sensitivity analysis. Visualization of cost function slices of (8), for the runs shown in Figs. 7–9. Only the first 10 min of the test are shown. (a) Trajectory cost function for NMPTC. (b) Evader/pursuer position difference cost function for NMPTC. Angle off tail ($\theta_t$) cost function for NMPTC. (d) State cost function for NMPTC.

The conditions to switch back to evader mode use the reverse conditions with a deadband to prevent cyclic switching behavior. Analytical bounds to prevent chattering are a subject of future research. The specific values used for these thresholds in these experiments are provided in Section VII-B.

## V. Development and Test Platforms

For the live demonstration, our vehicle had access to the adversary's state data at each timestep. Details regarding our vehicle's behavioral specifications, as well as the anticipated specifications of the adversary, are given throughout this section. We also provide the predictor function we used for our controller.

### A. Flight Aircraft

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. The T-33 included a third party autopilot system which did not include airspeed control of the aircraft. This aircraft will hereafter be referred to as the UAV. The UAV carried a safety pilot who could take control of the aircraft in the event of controller malfunction or poor decision making. Due to limitations on the autopilot system used for the controller interface, the safety pilot manually controlled the airspeed based on indicator alerts and a displayed target airspeed. The route and trajectory of the UAV was controlled by 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 UAV on a wireless link), was available via this avionics interface.

### 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 simulator (SILS) platform was provided by Boeing. This interface used Boeing's Open Control Platform (OCP), a black-box aircraft simulator called DemoSim and Java based UAV Experiment Controller GUI. The SILS versions of OCP and the Experiment Controller 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 controllers 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 UAV 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.

In order to test dynamic PEGs, Boeing modified the SILS to allow for two simulated UAVs to fly against each other, one as evader and the other pretending to be an F-15 pursuer. This allowed for extensive SILS testing of the NMPTC in PEGs with two simulated players.

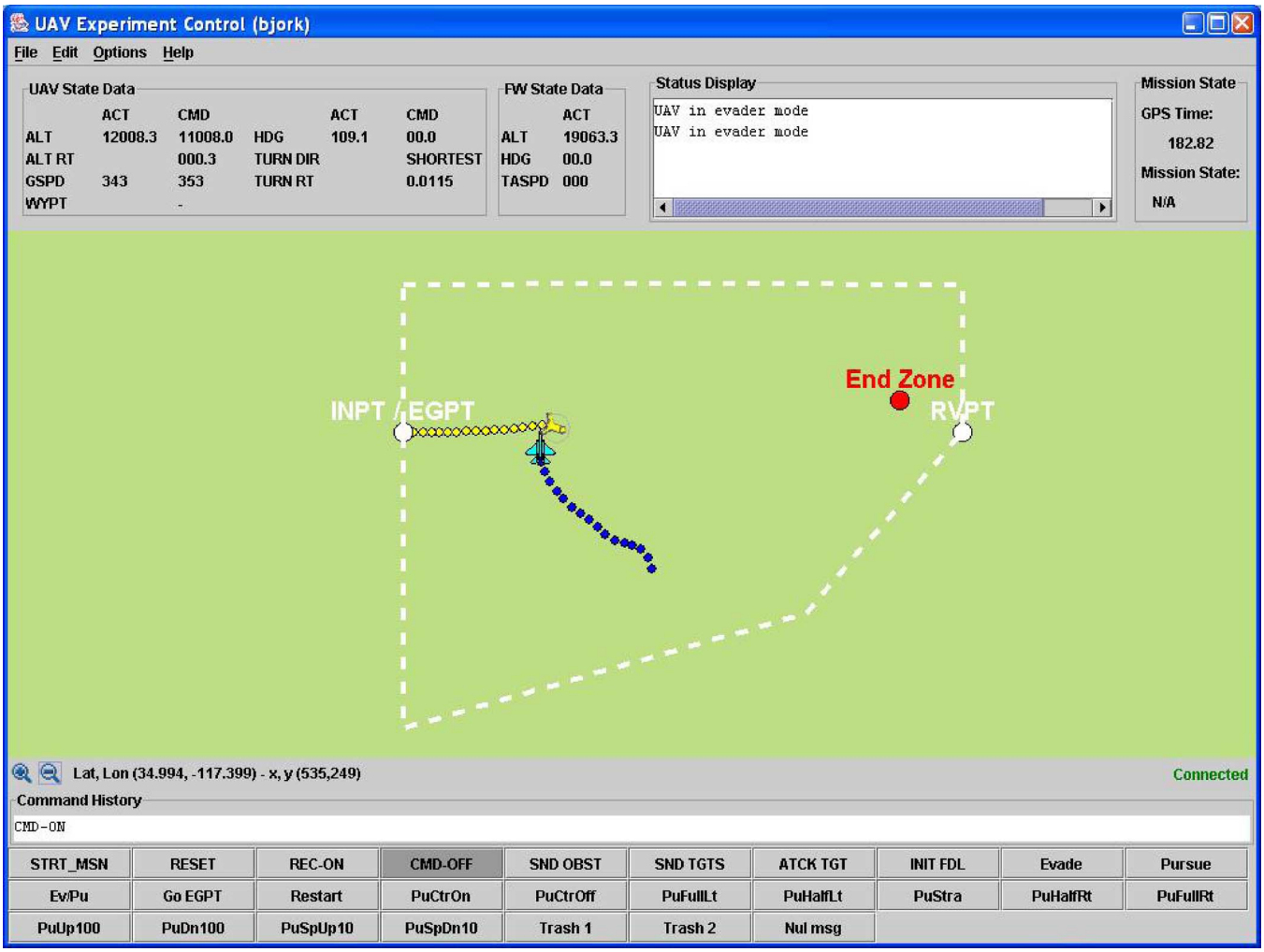


Fig. 11. UAV Experiment Controller: UAV is light gray, F-15 is dark.

### 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 to a proprietary aircraft simulation system. Along with software, the experiment test plans were provided with which Boeing developers tested and validated all the software for the various experiments, including the NMPTC controller for pursuit/evasion.

One of the major limitations on the HILS testbed was the lack of a human piloted pursuit aircraft. 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; thus 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 validated. The actual performance of the controller in a real PEG could only be evaluated during the actual flight test.

### D. Implementation

The NMPTC algorithm and the PEG 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 PEG. 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. Multiple iterations were required.

The pursuer algorithm was tuned to close on 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 this manner, as the final flight test of the system was with a trained pilot flying the pursuer aircraft as described below, the tuning of the cost functions was aimed at provided behavior similar to a real pilot, not an attempt to optimize the game outcome with respect to one player over the other. A discussion of the sensitivity of outcome to the cost function parameters is provided in Section VII-C.

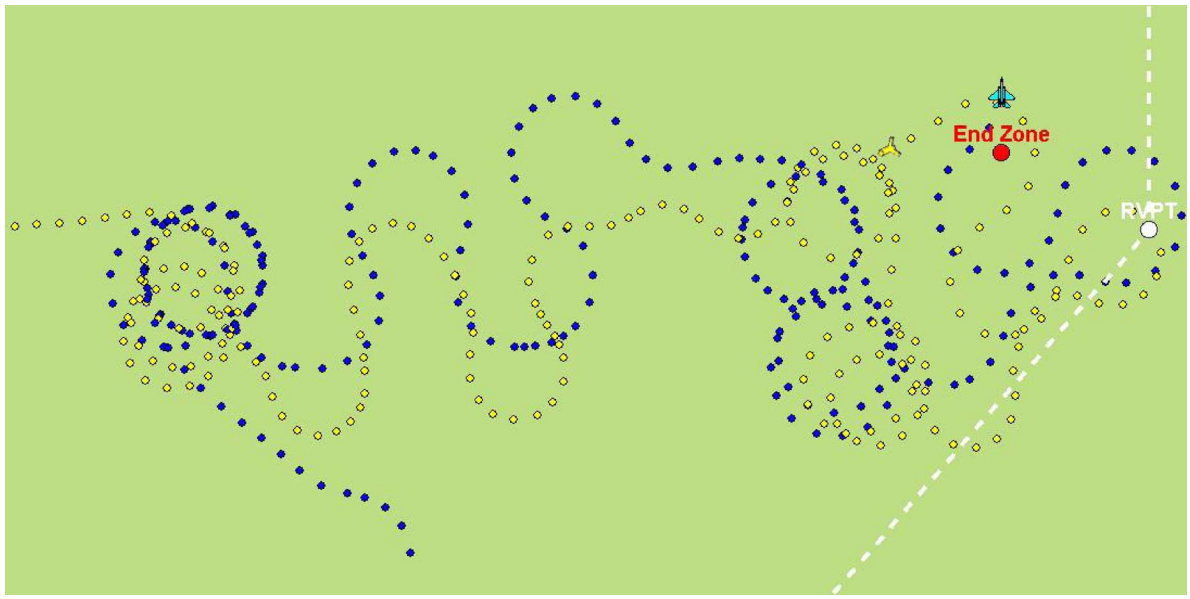


Fig. 12. 30-min simulation game: the pursuer (dark trace) started near the bottom, the evader (light trace) started on the left.

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 aircraft to verify the software for flight testing. In the flight test the other aircraft was an F-15 flown by a USAF-trained Boeing test pilot.

## VI. RESULTS OF SIMULATION AND DEMONSTRATION

### A. Simulation Results

In Fig. 7 the first half of a typical game result from simulation is shown. The evader enters in the northwest corner of the linearly defined play area, and its goal is to exit through the northeast corner of the play area. Simultaneously, the pursuer, who is on station in the southern region of the playing area, immediately begins to pursue the evader. In this particular result, the evader initially tries to bypass the pursuer and heads for the objective to the east; however when the pursuer approaches it turns toward the pursuer to avoid being in a targeting position. The remainder of the game includes descent and turning maneuvers, with the evader avoiding being targeted, but getting somewhat trapped near the lower (10 000 ft) altitude boundary. After 20 min, the game expires. In Fig. 8 two 2-D graphs provide more detailed views of the aircraft flight paths. The win conditions for this example are as discussed in Section III.

In Fig. 9 the $\theta_t$ and $\theta_n$ values from the perspective of the evading aircraft are shown, with a reference angle of 10°. Note that the evader's success is shown by keeping the $\theta_t$ and $\theta_n$ values from being below this line at the same time, right up until the 392 s mark when the pursuer managed to very briefly target the evader.

It can be observed from this result that the winning conditions are very difficult to achieve. For example, increasing the targeting range from 3 to 4 nm decreases the evader's success rate from 69.3% to 36.1%. However, these prescribed game conditions were used in the flight test experiments, and we retain them for our additional analysis.

The generated control inputs at each timestep were a direct result of the cost function at the previous timestep. For tuning the weights of individual portions of the cost function, we carefully selected subsets of the cost function and graphed them according to the prediction horizon and any strategies we selected. From another simulation result in which the pursuer successfully targeted the evader, Fig. 10(a) shows the cost function component for trajectory error ($\hat{\mathbf{y}}_k^{\mathrm{T}} Q \hat{\mathbf{y}}_k$, for $N$ steps, varying by the simulation time). Large values indicate undesired cost, due ostensibly to deviations from the trajectory to avoid loss by targeting conditions. The predicted cost function value, calculated at each time step, can be sliced across the "Simulation Time" axis to see the predicted values of this cost element based on the selected strategy at that timestep. These cost functions are the collection of all control inputs for the first 10-min of simulation time, and a 30-step lookahead is used.

Another set of interesting plots deserve some special discussion. For example, in Fig. 10(b) the cost function of proximity of the evader and pursuer is shown. Fig. 10(c) shows the separate cost of $\theta_t$, which the evader should use to "shake" the pursuer off its tail. The cost of violating constraints on the evader's state are shown in Fig. 10(d), in which the controller considers limitation on speed, altitude, etc. These are for the same example used above and shown in Figs. 7–9.

These plots also provide an ability to see how this particular game developed in time. After 5 minutes, there were high costs for trajectory, which indicates that the evading vehicle was either threatened, or perceived a threat. This can be verified by the spikes in proximity difference around 2–3 min; in $\theta_t$ a brief spike is observed; more significant spikes occur around 5 min

when the evader manages to avoid being targeted; and finally at about 6.5 min when the evader is targeted by the pursuer. The errors in state were fairly low overall, and only anticipated errors in state (shown in the "Lookahead" axis) are seen with a high cost, except at the very start of the experiment when the aircraft is getting on track from its initial conditions.

### *B. Simulation in the Experiment Controller*

The experiment controller used in the simulation and flight test experiments is shown in Fig. 11 at the start of a SILS experiment. In this case the evader has entered 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. 12 shows the outcome of this experiment after about 30 min of real time (experiments were often allowed to continue past the 20 min PEG rules to further evaluate the controllers). 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.

### *C. Flight Test Validation*

The NMPTC controller was provided to Boeing 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 Edwards AFB during the weeks of June 14–25, 2004. This experiment was run four 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, and as in the HILS it could only follow a preprogrammed flight path. In these cases the simulated F-15 flew 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, though the UAV executed evasive maneuvers for several minutes. In this experiment, the F-15 used its speed advantage to great benefit.

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. 13, the paths of the aircraft can be seen. The F-15 was carrying out a search pattern as the UAV approached, and in this experiment the F-15 flew at the same velocity as the UAV.

During the flight demonstration, the UAV detected an advantageous condition and switched to pursuer mode and successfully targeted the F-15. The experiment was allowed to continued after this (not shown) and the F-15 did shake the UAV

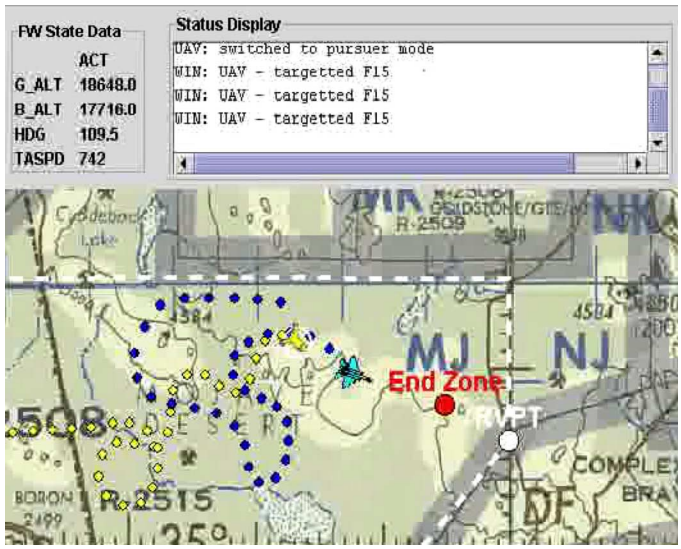


Fig. 13. Flight test experimental results: A symmetric PEG in which the UAV (light trace) has just targeted the F-15 (dark trace).

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 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).

## VII. Simulation Analysis

After our successful flight demonstration, we performed regressive analysis of the pursuer/evader controllers through structured variation of the initial conditions of the game. We used the same boundary conditions found in the live demonstration, and the DemoSim executable as the dynamics for the vehicle. These results fall into two categories: those in which we do not permit switching by the evader to pursuer mode, and those which we permit the evader to change modes in order to target the pursuer.

### *A. Unswitched PEGs*

The results of these simulations are shown in Fig. 14. The winning vehicle is shown as an "$o$", and the losing vehicle as an "$x$". The initial velocity vector is indicated by the quiver extending from the initial $(x, y)$ position, and a line connects the ends of these quivers to correlate which points belong to a single simulation.

While the two aircraft have identical performance characteristics (due to the restrictions place by the DemoSim simulation), an advantage could be given to either the pursuer or evader through the dynamically accurate modeling characteristics of DemoSim as follows: we operate one vehicle at a higher altitude, which the dynamics engine accurately simulates with less air density, and thus lower maneuvering performance but higher speed. This higher altitude does not give a tactical height advantage to that vehicle, however, as the relative height difference was compensated for in the use of the shared state information

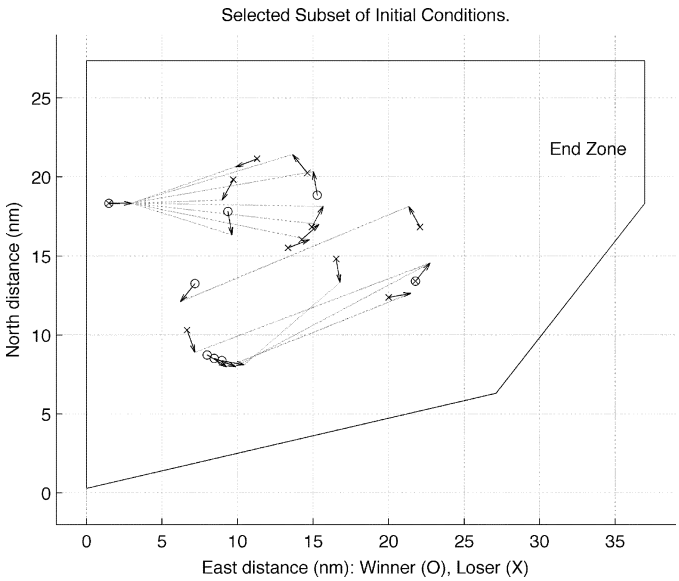


Fig. 14. View of initial conditions from above, with winner as "$o$", loser as "$x$". For clarity, a reduced set of simulations were chosen. The dashed lines connect evader and pursuer starting positions for each test run, and arrows indicate initial heading of each vehicle.

and the NMPTC cost functions. Thus, the vehicles believed they were operating at the same altitude regardless of this offset.

An important note about the simulation results is in regard to the nondeterministic behavior of the simulation itself. The SIL simulation system includes four separately initiated program processes, each of which have significant start up times (on the order of several second). Communication between the processes occurs through middleware, at data rates of approximately 1 Hz. During startup, vehicles must move to maintain flight, and if no data is received from the other vehicle, it cannot be used as part of the predictive controller. The longer it takes to communicate with the other vehicle, the more the pursuer (and evader) will change their initial position and heading. Thus, the jitter in establishing communication serves to locally randomize starting *game* conditions, and it may directly affect individual outcomes, despite identical starting *simulation* conditions.

In all, we ran 335 simulations with no switching permitted by the evader. Table I shows the percentage win and loss by evader/pursuer in these experiments. In the event that there was no clear winner within the time limits, a win was granted to the evader, per the rules of the game. Here we see that there is a bias towards the evader, which is justifiable as the original objective of the NMPTC controller was to focus on evasion within the predefined rules of the PEG. This focus governed the development of the controller that was validated in the flight test experiments and has not been modified for these additional experiments.

These results were also examined to see the effect of initial conditions on the outcomes, and Fig. 14 shows two distinct sets of initial conditions. The first (upper) set of results begin with the evader near the ingress point and the pursuer at a random point around a circle in front of the evader. The dashed lines connect the evader and pursuer starting positions to indicate each test case. The second (lower) set of results show both aircraft at a random point around their own circles, with the pursuer again between the evader and the objective end point, labels the "End Zone".

TABLE I
SUMMARY OF WIN/LOSS OUTCOMES OF SIMULATIONS IN FIG. 14

| | **Count** | **Likelihood (Pct)** |
|---|---|---|
| Evader Wins | 232 | 69.2% |
| Pursuer Wins | 103 | 30.8% |
| Total Runs | **335** | |
| **Event Type** | **Runs w/ Evt** | **Likelihood (Pct)** |
| Evader Reaches End Zone | 72 | 21.5% |
| Pursuer Targets Evader | 103 | 30.8% |
| Evader Wins due to Time Elapsed | 160 | 47.7% |

TABLE II
SUMMARY OF WIN/LOSS OUTCOMES OF SIMULATIONS WITH CHANGES IN THE WINNING CONDITIONS

| **Angle** | **Distance** | **Evader wins (Pct)** |
|---|---|---|
| 10° | 3 $nm$ | 69.2% |
| 15° | 3 $nm$ | 47.8% |
| 10° | 4 $nm$ | 36.1% |

TABLE III
SUMMARY OF WIN/LOSS OUTCOMES OF SIMULATIONS WHERE THE EVADER IS PERMITTED TO SWITCH TO "PURSUER" MODE

| | **Pursuer Wins** | **Evader Wins** | | **Total** |
|---|---|---|---|---|
| | Targets Evader | Time | End Zone | |
| 1. Adv. Pursuer w/ Eq. 9 | 11 | 13 | 2 | 26 |
| 2. Adv. Pursuer w/ Eq. 10 | 11 | 20 | 0 | 31 |
| 3. Even w/ Eq. 9 | 8 | 18 | 2 | 28 |
| 4. Even w/ Eq. 10 | 3 | 17 | 3 | 23 |
| 5. Adv. Evader w/ Eq. 9 | 8 | 16 | 4 | 28 |
| 6. Adv. Evader w/ Eq. 10 | 1 | 26 | 0 | 27 |

The rules of the game are also a critical factor in determining the outcome of the PEGs, and while the original rules were set for the flight test experiments and used throughout the development and testing of the NMPTC controller, it was possible to investigate the effect of these rules on the outcomes. As noted earlier and seen in Fig. 9, the pursuer is often *almost* able to target the evader. By way of post-processing the results with different winning conditions, as shown in Table II, the win percentage of the evader drops dramatically if targeting conditions are relaxed. This is true to a point; however it also becomes easier for the evader to target the pursuer if the conditions are kept symmetric and it is allowed to switch modes.

### B. Switched PEGs

Following the evaluation of the non-switching experimental results, a specific starting position with roughly average chances of success (66.8% for the evader in these results compared to 69.2% for the full set of results presented in the previous section) was used to evaluate the effect of allowing for mode switching for the UAV between "evader" and "pursuer" mode. During these experiments, which permit the evader to switch to "pursuer" mode, we ran a total of 164 simulations, summarized in Table III. The results clearly show that if the UAV is allowed to switch to "pursuer" mode, it can extend its chances of survival significantly. If those conditions are further relaxed, chances are further improved.

Recalling the switching controller from Fig. 6, we provide the values of $\epsilon_i$ in that figure. Our first switching conditions, called the *tight* conditions in the table, are

$$\begin{aligned} \text{Ev} \rightarrow \text{Pu} &= (|\theta_{n_e}| < 60) \wedge (|\theta_{n_p}| > 90) \wedge (d < 6) \\ \text{Pu} \rightarrow \text{Ev} &= (|\theta_{n_e}| > 80) \vee (d > 10) \end{aligned} \tag{9}$$

where angles are specified in degrees, and distance in nautical miles. With these conditions, the likelihood of UAV survival increases over the likelihoods presented in Table I. If we use the following switching conditions:

$$\begin{aligned} \text{Ev} \rightarrow \text{Pu} &= (|\theta_{n_e}| < 90) \wedge (|\theta_{n_p}| > 45) \wedge (d < 6) \\ \text{Pu} \rightarrow \text{Ev} &= (|\theta_{n_e}| > 150) \vee (d > 10) \end{aligned} \tag{10}$$

called the *easy* conditions in Table III, we see that the evader's performance improves dramatically. Again, angles are specified in degrees, and distances in nautical miles.

This improvement in the results for the evader can be interpreted as with the relative match in performance of the two aircraft, when both are in "pursuer" mode there is an even (and fairly low) chance that one will target the other, hence, the longer the UAV stays in "pursuer" mode, the higher the odds that it will remain in the game until the end at time $T$.

### C. Sensitivity Analysis

As explained in Section V-D above, the cost function parameters used in these experiments were meant to produce reasonable flight behavior similar to what would be expected from a real pilot, which was successfully achieved as noted in Section VI-C. However, it is naturally of interest to determine an optimal set of cost functions to achieve one outcome over another, e.g., the evader winning the game at the expense of the pursuer. This is left largely for future work, however an initial analysis and discussion is provided here.

To evaluate the sensitivity of the outcome to the cost functions used in the NMPTC algorithm, the effect of the angle off tail cost function [i.e., the $\mathbf{d}$ for the $G$ matrix in (8)] on the pursuer's winner percentages was studied through simulation experiments. This function was chosen as it directly affects the evasive actions taken by the evader. A set of 20 new test results were conducted with the same initial conditions and the maneuvering advantage given to the evader. Then $\mathbf{d}$ was scaled to increase the relative effect of the angle off tail cost function by a factor of four and then by eight and the experiment was repeated with the same initial conditions in each case. The result was an increase in the pursuer's winning percentages from 60% with the baseline case to 70% and 75% for the latter cases, respectively.

This demonstrates that there is considerable opportunity to improve the winning outcomes for either the pursuer of the evader through cost function tuning. For the simulations presented in Table III and Fig. 14, the fielded flight demonstration cost function was preserved in order to demonstrate that the results from the live flight experiment were in line with expected outcomes, and not just a chance occurrence. We leave further results in the experimental and analytical optimization of the cost functions to future work.

### D. Baseline Analysis

In addition to the validation of the simulation methods with the flight test, the algorithm was compared to four baseline tests to validate the following assumptions:

- the pursuer can likely catch an unaware evader;
- the pursuer can likely catch a naive evader;
- the evader can reliably avoid an unaware pursuer;
- the evader can reliably avoid a naive pursuer.

This led to the following baseline experiments to validate the approach:

1) the evader flies directly to the end zone, and is not given access to the pursuer's state;
2) the evader flies a figure-8 maneuver with no access to the pursuer's state;
3) the pursuer flies a figure-8 maneuver, with no access to the evader's state unless the evader flies into the pursuer's cone in any orientation (as shown in Fig. 2);
4) same as the previous, except that the evader can switch to pursuer mode.

In each experiment where one of the opponents was flying a figure-8, the other vehicle was restricted to the same altitude. For each experiment 26 flight tests were performed to establish the rates of success.

*1) Baseline Experiment 1—Unaware Evader:* In this experiment, the evader ignores any pursuer data and flies directly toward the end zone. Experiments where the pursuer began in an advantageous or neutral initial condition resulted in 100% pursuer wins. In cases where the pursuer was clearly at a disadvantage due to initial position and orientation, the evader was able reach the end zone. When sampled over the entire space, the pursuer won 73% of the time.

*2) Baseline Experiment 2—Naive Evader:* In this experiment, the evader flies a figure-8 path in the middle of the playing space and the pursuer has full access to evader state. Here the pursuer was successful in 92% of the cases with the evader managing to win through elapsed time in cases where the pursuer is at a disadvantage due to initial conditions.

*3) Baseline Experiment 3—Unaware Pursuer:* In this case, the pursuer flies in a figure-8 pattern as a test to ensure that the evader algorithm would successfully avoid the pursuer and complete the mission of reaching the End Zone. In this case the evader won by reaching the End Zone in 100% of the matches. Note that none of the experiments placed the evader within the targeting criteria as an initial condition.

*4) Baseline Experiment 4—Naive Pursuer:* This final baseline experiment uses the same conditions as Baseline Experiment 3, except that the evader was able to switch to the pursuer mode if an advantage was detected. In this case, the evader still won 100% of the games, however reaching the End Zone accounted for only 23% of the wins. The other wins came from time elapsed for (54%) and targeting the pursuer (23%). An example of the last result is shown in Fig. 16. Similarly, these results can be accounted for in that when the evader choses to switch to a pursuer mode it may have less chance of reaching the End Zone, however it does not jeopardize its chances of winning due to the expiration of game time.

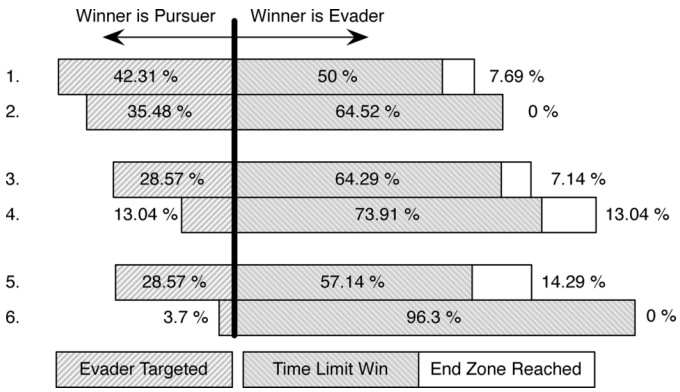


Fig. 15. Graphical summary of Table III. Note that the easy switching conditions always improve the evader's behavior over the original switching conditions.

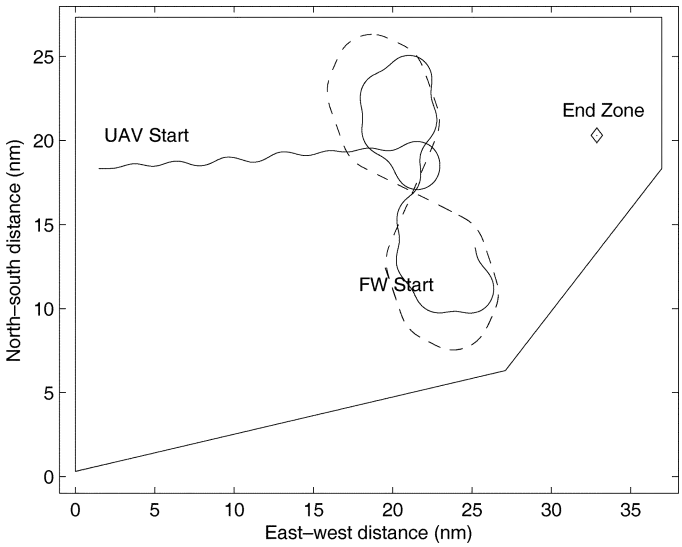


Fig. 16. Baseline experiment in which the pursuer flies a figure-8 pattern (counterclockwise at the bottom and clockwise at top of figure-8) and the UAV is able to switch to pursuer mode on its way to the End Zone. The evader does so and is successful in targeting the pursuer.

### *E. Comparison With Related Work*

These results show that an evader strategy of a simple evasive maneuver (figure-8) loses the vast majority of the time to a pursuer equipped with the predictive controller. When the evader succeeds, it is due to a clear disadvantage in initial conditions for the pursuer. In cases where the evader is unaware it is being targeted, the evader loses 74% of the time, with wins coming when there is a clear disadvantage for the pursuer in initial conditions (i.e., the evader is much closer to the end zone than it is to the pursuer).

When the pursuer has a naive (or no) strategy, the evader clearly has the advantage, winning in every case, either by reaching the end zone, or by elapsed time. As was shown in the earlier results, if the pursuer has knowledge of the evader, then initial conditions may determine the winner, but with no knowledge by the pursuer of the evader's state, the evader can only lose by blundering into the targeting criteria zone.

Some discussion in merited with regards to how existing approaches from Section II-D might fare in these scenarios. Most of those approaches are not suited for these scenarios, with the following exceptions. The probabilistic threat exposure map [30] is relevant to Baseline Experiment 1, where state data of the pursuer are unavailable to the evader, and the construction of the threat exposure map would be an interesting topic, as in order to stay within the necessary boundaries, the boundaries would need to be encoded as threats. Further, the construction of the map would be ill-posed in some sense, as the location of the pursuer would be unknown after some time. Thus some prediction is still required, based on the estimated strategy of the pursuer.

The probabilistic games in a plane [28] and the single decision point for evasive maneuver [25] are relevant to Baseline Experiment 2, where the decision for which maneuver to perform (without further knowledge of the state of the pursuer) is important. Each of these approaches may provide an improvement in the success rate that was observed in Baseline Experiment 2 by selecting a different evasive maneuver to follow, based on particular knowledge of the pursuing vehicle. However, if the pursuer can predict the evader's strategy, then even this naive approach may not succeed.

## VIII. Future Work

Some significant portions of the framework are reusable, and perhaps subject to parameterization on a higher level. Development of a computational, real-time guaranteed, platform-independent MPC framework is an ongoing area of interest for the authors. We saw some success in the reuse of the algorithms across platforms from rotorcraft [31], [34] to fixed-wing aircraft [1], [2], although the implementation for our fixed-wing example was rewritten from published equations, not from existing code.

We did see software codebase reuse when the Sydney-Berkeley Driving Team [35] used the software from this flight demonstration (i.e., fixed-wing air vehicles) after substituting the kinematic model of a four-wheeled, front steering ground vehicle. This required, of course, rewriting the cost equations since the game no longer involved a pursuer. Further dividing the portions of the generic infrastructure into a reusable computational toolbox is in some sense an exercise in abstraction, though we expect that research tasks involving anytime results from various optimizers, heuristics for optimization (such as the alternative path planner used in [36]) including simulated annealing and genetic algorithms, to pose interesting investigation avenues for selecting the appropriate controller based on the evaluation of performance metrics such as time, memory, and game conditions satisfaction.

An interesting further computational avenue is the introduction of adaptive controllers to attempt to enforce the predictive model's behavior, regardless of the dynamics involved. For air vehicles, this could allow the kinematic model to be supplemented by this adaptive module, which could enable the traces to be more reflective of actual behavior, and thus allow the predictive paths to be much closer to actual paths.

Undoubtedly, the most time-consuming portion of the system development is the tuning of the cost function. In order to more closely represent human behaviors, the use of offline computational methods such as those by Raffard *et al.* [37], to globally

optimize the parameters of the cost function based on *desired* behavior of some initial conditions would enable human-based training of the predictive controller. This would permit, for example, a combat pilot to record behavior in the simulator over several months, and then push this information into the cost function of the NMPTC controller.

As demonstrated in Section VII-C, there is considerable opportunity to improve the outcomes in favor of one participant over the other, and this would perhaps be most effectively achieved in a competitive forum where the pursuer and evader algorithms are developed separately, or in which the pursuer is piloted in the simulation environment by a trained pilot.

There are interesting applications to this for ground vehicles as well, such as autonomous driving controllers for automobiles, that can emphasize smooth steering, smooth acceleration, etc., based on learning a driver's preferences—achieving personalization significantly more advanced than seat depth and mirror position.

## IX. Conclusion

In this paper we have presented experimental results of the effectiveness of the nonlinear model-predictive tracking controller approach to a 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 evader mode to a UAV operator, it is possible for a remote operator to relinquish control of the vehicle in time-critical situations, allowing 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 PEGs with a piloted F-15. NMPTC had not yet been demonstrated on full-scale fixed-wing aircraft for the pursuit/evasion problem, and this work shows that this method is appropriate when providing input to an autopilot interface.

## Acknowledgment

This work benefited greatly from previous work by Dr. D. H. C. Shim at UC Berkeley, and Dr. H. J. Kim during the project performance. The authors would also like to thank B. Mendel, J. Rosson, Dr. J. L. Paunicka, and Dr. D. E. Corman of Boeing Phantom Works, who provided the T-33 testbed, and facilitated the demonstration in June 2004.

## References

[1] J. Sprinkle, J. Eklund, H. Kim, and S. Sastry, "Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller," in *Proc. 43rd IEEE Conf. Decision Control (CDC)*, Dec. 2004, vol. 3, pp. 2609–2614.

[2] J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit evasion games on a fixed wing aircraft," in *Proc. Amer. Control Conf. (ACC)*, 2005, pp. 1509–1514.

[3] AFMC Public Affairs, Wright-Patterson AFB, OH, "Global HAWK UAV supports OEF recon," AFMC News Service Release, Dec. 2002.

[4] I. Yavrucuk, S. Unnikrishnan, and J. Prasad, "Envelope protection for autonomous unmanned aerial vehicles," *J. Guid., Control, Dyn.*, vol. 32, no. 1, pp. 262–275, 2009.

[5] DARPA, Arlington, VA, "DARPA completes autonomous airborne refueling demonstration," DARPA Press Release, Aug. 2007.

[6] M. L. Fravolini, A. Ficola, G. Campa, M. R. Napolitano, and B. Seanor, "Modeling and control issues for autonomous aerial refueling for UAVs using a probe-drogue refueling system," *Aerosp. Sci. Technol.*, vol. 8, no. 7, pp. 611–618, 2004.

[7] S. Ross, M. Pachter, D. Jacques, B. Kish, and D. Millman, in *Proc. IEEE Aerosp. Conf.*, Jul. 2006.

[8] G. Warwick, *Boeing to lead UAV aerial refueling demo*, ser. Aviation Week. New York: McGraw Hill, Nov. 2008.

[9] T. Lam, M. Mulder, and M. V. Paassen, "Haptic feedback in uninhabited aerial vehicle teleoperation with time delay," *J. Guid., Control, Dyn.*, vol. 31, no. 6, pp. 1728–1739, 2008.

[10] A. Richards and J. How, "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility," in *Proc. Amer. Control Conf.*, 2003, pp. 4034–4040.

[11] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *Proc. Amer. Control Conf.*, 2004, pp. 5576–5581.

[12] J. Maciejowski, *Predictive Control with Constraints*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[13] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "User's Guide for NPSOL 5.0: A fortran package for nonlinear programming," Syst. Opt. Lab., Dept. Operat. Res., Stanford Univ., Stanford, CA, Tech. Rep. SOL 86-2.

[14] K. Holmström, M. M. Edvall, and A. O. Göran, "TOMLAB-for largescale robust optimization," presented at the Nordic MATLAB Conf., Copenhagen, Denmark, 2003.

[15] H. Chung, E. Polak, and S. Sastry, "An external active-set strategy for solving optimal control problems," *IEEE Trans. Autom. Control*, vol. 54, no. 5, pp. 1129–1133, May 2009.

[16] D. Shim, H. Chung, and S. Sastry, "Conflict-free navigation in unknown urban environments," *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 27–33, Sep. 2006.

[17] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.

[18] M. Zeilinger, C. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization," in *47th IEEE Conf. Dec. Control (CDC)* , 2008, pp. 4718–4723.

[19] U. Maeder, R. Cagienard, and M. Morari, "Explicit model predictive control," in *Advanced Strategies in Control Systems with Input and Output Constraints*. Berlin/Heidelberg, Germany: Springer, 2007, vol. 346, Lecture Notes in Control and Information Sciences, pp. 237–271.

[20] T. Başar and G. J. Olsder, *Dynamic Non-Cooperative Game Theory*, 2nd ed. San Diego, CA: Academic Press, 1995.

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

[22] R. Isaacs, *Differential Games*. Hoboken, NJ: Wiley, 1967.

[23] S. LaValle and S. Hutchinson, "Game theory as a unifying structure for a variety of robot tasks," in *Proc. IEEE Int. Symp. Intell. Control*, 1993, pp. 429–434.

[24] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D., Scientific Comput. Computat. Math. Program, Stanford Univ., Stanford, CA, 2002.

[25] S. Le Menec, "Differential games and symbolic programming to calculate a guaranteed aircraft evasion in modern aerial duels," in *Proc. 33rd IEEE Conf. Dec. Control*, 1994, pp. 3868–3870.

[26] R. L. Shaw, "Fighter combat: Tactics and maneuvering," United States Naval Inst., Annapolis, MD, 1985.

[27] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 662–669, Oct. 2002.

[28] J. Hespanha, M. Prandini, and S. Sastry, "Probabilistic pursuit-evasion games: A one-step Nash approach," in *Proc. 39th IEEE Conf. Dec. Control*, 2000, pp. 2272–2277.
[29] R. Vidal and S. Sastry, "Vision-based detection of autonomous vehicles for pursuit-evasion games," presented at the IFAC World Congr. Autom. Control, Barcelona, Spain, 2002.
[30] U. Zengin and A. Dogan, "Real-time target tracking for autonomous UAVs in adversarial environments: A gradient search algorithm," in *Proc. 45th IEEE Conf. Decision Control*, 2006, pp. 697–702.
[31] H. Kim, D. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," , vol. 5, pp. 3576–3581, 2002.
[32] "Progress in Systems and Control Theory," in *Nonlinear Model Predictive Control*, F. Allgöwer and A. Zheng, Eds. Basel-Boston-Berlin: Birkhäuser Verlag, 2000, vol. 26.
[33] G. J. Sutton and R. R. Bitmead, "Computational implementation of nonlinear predictive control on a submarine," in *Nonlinear Model Predictive Control*. Basel- Boston-Berlin: Birkhäuser Verlag, 2000, vol. 26, Progress in Systems and Control Theory, pp. 461–471.
[34] D. Shim, H. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *Proc. 42nd IEEE Conf. Decision Control*, 2003, pp. 3621–3626.
[35] J. Sprinkle, J. M. Eklund, H. Gonzalez, E. I. Grøtli, B. Upcroft, A. Makarenko, W. Uther, M. Moser, R. Fitch, H. Durrant-Whyte, and S. S. Sastry, "Model-based design: A report from the trenches of the DARPA Urban Challenge," *Softw. Syst. Model.*, vol. 8, no. 4, pp. 551–566, 2009.
[36] B. Upcroft, A. Makarenko, M. Moser, A. Alempijevic, A. Donikian, W. Uther, and R. Fitch, "Empirical evaluation of an autonomous vehicle in an urban environment," *J. Aerosp. Comput., Inf., Commun.*, vol. 4, no. 12, pp. 1086–1107, Dec. 2007.
[37] R. Raffard, K. Amonlirdviman, J. Axelrod, and C. Tomlin, "Automatic parameter identification via the adjoint method, with application to understanding planar cell polarity," in *Proc. 45th IEEE Conf. Decision Control*, 2006, pp. 13–18.

**J. Mikael Eklund** (S'98–M'03) received the B.Sc. and M.Sc. degrees in engineering and the Ph.D. degree from Queen's University, Kingston, ON, Canada, in 1989, 1997, and 2003, respectively.

He is an Assistant Professor and Program Director for the Department of Electrical and Software Engineering, University of Ontario Institute of Technology, Oshawa, ON, Canada. Until August 2006, he was a Visiting Postdoctoral Scholar with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. His research includes the area of autonomous systems including robotic vehicles and in smart medical systems. Between his Bachelor and Masters studies he was a Simulator Flight Control Systems Engineer with CAE Electronics in Montreal, QC, Canada.

**Jonathan Sprinkle** (S'96–M'03) received the B.S.E.E. degree *in cursu honorum*, *cum laude* from Tennessee Technological University, Cookeville, in 1999, where he was the first graduate of the Computer Engineering program, and the first electrical engineering double major, the M.S. degree in 2000, and the Ph.D. degree from Vanderbilt University, Nashville, TN, in 2003.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, University of Arizona, Tempe. In 2009, he received the UA's Ed and Joan Biggers Faculty Support Grant for work in autonomous systems. Until June 2007, he was the Executive Director of the Center for Hybrid and Embedded Software Systems, University of California, Berkeley. His research includes the area of intelligent autonomous systems, through building blocks of domain-specific modeling, metamodeling, and generative programming. He was the co-Team Leader of the Sydney-Berkeley Driving Team, a DARPA Urban Challenge collaboration with partners Sydney University, University of Technology, Sydney, and National ICT Australia.

**S. Shankar Sastry** (S'79–M'80–SM'95–F'95) received the M.A. degree (*honoris causa*) from Harvard University, Cambridge, MA, in 1994 and the Ph.D. degree from the University of California, Berkeley, in 1981.

He is the Dean of Engineering with University of California, Berkeley. He was on the faculty of Massachusetts Institute of Technology as an Assistant Professor from 1980–1982 and Harvard University as a chaired Gordon McKay Professor in 1994. His areas of personal research are embedded and autonomous software for unmanned systems (especially aerial vehicles), computer vision, computation in novel substrates such as quantum computing, nonlinear and adaptive control, robotic telesurgery, control of hybrid and embedded systems, network embedded systems and software. He has supervised over 50 doctoral students to completion and over 50 M.S. students. His students now occupy leadership roles in several locations and on the faculties of many major universities in the United States and abroad. He has coauthored over 400 technical papers and 9 books.

Prof. Sastry was a recipient of the President of India Gold Medal in 1977, the IBM Faculty Development Award for 1983–1985, the NSF Presidential Young Investigator Award in 1985, the Eckman Award of the American Automatic Control Council in 1990, the Ragazzini Award for Distinguished Accomplishments in teaching in 2005, the distinguished Alumnus Award of the Indian Institute of Technology in 1999, the David Marr prize for the Best Paper at the International Conference in Computer Vision in 1999, and an honorary doctorate from the Royal Swedish Institute of Technology in 2007. He is a member of the National Academy of Engineering and the American Academy of Arts and Sciences (AAAS).