

LEENA SINGH
JOHN PLUMP
MARC MCCONLEY
BRENT APPLEBY



SOFTWARE-ENABLED CONTROL: AUTONOMOUS AGILE GUIDANCE AND CONTROL FOR A UAV IN PARTIALLY UNKNOWN URBAN TERRAIN

Copyright © 2003 American Institute of Aeronautics and Astronautics, Inc. Based on a paper presented at the Guidance, Navigation, and Control Conference and Exhibit, Austin, TX, August 11-14, 2003.

ABSTRACT

Draper Laboratory has developed a dynamic model-based, hybrid control theoretic guidance approach to autonomously design agile, realizable trajectories for unmanned vehicles. This hybrid control-based method synthesizes the guidance trajectory by optimally combining trajectory elements from a stored database of trim and agile flight maneuver primitives. An off-line computation step in the guidance logic computes and stores the maneuver primitives in a maneuver library; these primitives span a discrete, reachable subspace of maneuvers of the particular aircraft and include benign and aggressive (but still feasible) maneuvers. The approach has been presented previously.^{[1],[2]} In this paper, we present some details of the method to define and implement (in real-time) three-dimensional (3D) maneuver bases. We also present simulation and experimental results of using the hybrid maneuver selection logic to design trajectories in 3D space. We also introduce an approach to effect constraint satisfaction for collision avoidance during the maneuver synthesis process due to unknown threats in the urban flight terrain such as utility lines and trees. We will present our urban mission scenario and show simulation results using our hybrid maneuver logic to design the guidance trajectory to fly a low-altitude mission without colliding with buildings, wires, or trees in that urban landscape. The hybrid logic is being integrated into the flight control system developed by the Georgia Institute of Technology and is based on the Open Control Platform.

INTRODUCTION

In this paper, we describe a hybrid logic automaton-based guidance algorithm that solves a minimum-time motion planning problem onboard an autonomous vehicle's flight control system to synthesize agile trajectories in 3D space in a cluttered urban terrain. The onboard trajectory planner sequences maneuvers and trims from a stored library of agile maneuvers and trim flight primitives to synthesize a dynamically feasible, time-optimal trajectory with state feedback. Key aspects of the algorithm are that it synthesizes aggressive trajectories inspired by the dynamic capabilities and limits of the aircraft, and that it embodies feedback to synthesize guidance solutions and flight control commands in real-time. Under the Defense Advanced Research Projects Agency (DARPA) Software-Enabled Control (SEC) program, we are extending this hybrid automaton agile trajectory design method with collision constraints to enable autonomous missions where the guidance logic must synthesize trajectories in urban terrain to overcome dangers in low-altitude flight such as unknown utility lines.

To use this hybrid logic maneuver sequencing approach, we first populate maneuver and trim libraries with a discrete set of trajectory primitives that parameterize the vehicle’s dynamic capabilities. Trim primitives are defined in a reduced, spatially invariant trajectory space parameterized by $(V, \alpha, \beta, \gamma)$. In our notation, V is speed, α is angle of attack, γ is the flight path angle, and β is sideslip. A series of dynamically feasible minimum time transitions between all these trim conditions constitutes the library of maneuver primitives. We then set up a dynamic programming problem; the motion planner solves this dynamic programming problem by optimally sequencing various trim and maneuver elements to construct the solution trajectory. We will review these elements of the hybrid automaton first before we describe how we use it to synthesize collision-avoiding trajectories in an urban terrain.

The paper format is as follows: The next section, “Problem Definition,” describes the problem and the layout. Next, in “Summary of Hybrid Maneuver Logic,” we summarize the hybrid logic for trajectory design. In “Obstacle Avoidance,” we describe our approach to avoiding collisions. Finally, in “Description of the Experiments,” we describe the final scenario that we will implement and our preliminary results to date after the software integration with the DARPA SEC rotorcraft platform software.

PROBLEM DEFINITION

Next-generation autonomous aircraft are being investigated for missions in low-altitude applications and in partially unknown terrain such as urban areas where survivability threats predominate. Military research and procurement are interested in low-altitude flight because it allows the vehicle to fly below the radar deck to escape detection; flying in the ground clutter below tree-top level and between buildings provides further masking capability from other detection sources. As a result, low-altitude flight holds promise for improving survivability from detection and tracking systems. In addition, the vehicle must be able to fly fast as well, because high speeds provide the vehicle a further protection from detection and tracking. Figure 1 shows the relationship of survival probabilities against detection and tracking. The figure indicates that high-speed, low-altitude flight provides the best chance for survival against a hostile enemy.

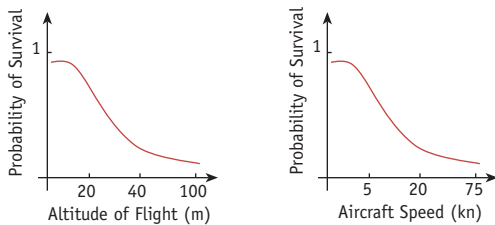


Figure 1. Plot showing probabilities of survival as functions of altitude and speed.

In military missions in hostile urban terrain, the vehicle must be able to operate very agilely and maneuver rapidly to improve survivability against colliding with low-lying obstacles. Threats to survival in urban terrain include more than just hostile intent; utility lines are an ever-present danger to low-flying aircraft and are nearly impossible to map *a priori*. Therefore, the aircraft must be able to detect them during flight and react rapidly. However, the range of the obstacle detection sensor parameterizes the agility required from the aircraft—the more agile the aircraft, the lower the necessary sensor operating range for safe flight performance. Similarly, the more agile the aircraft, the faster it can operate in the cluttered terrain for a given sensor range. Small, stealthy unmanned aerial vehicles (UAVs) will not be able to carry the substantial payload associated with long-range, highly collimated laser systems, hence the appeal of smaller, lighter sensor scanners. With conventional autonomous guidance and control approaches, the lower range of the smaller portable scanners implies that the aircraft will have to operate at very low speeds so that it has the dynamic range to detect, plan, and execute a trajectory that avoids utility lines. Therefore, while new and agile vehicles are being designed, the improved mechanical agility will be rendered useless without new guidance strategies that can call on and deploy the higher maneuvering capabilities of the vehicle. With our autonomous guidance and control algorithm that synthesizes aggressive command trajectories for the flight controller, we plan to use our high-agility trajectory design methods with a feasible, lightweight sensor, and design trajectories that use the physical capabilities of the aircraft instead of being limited to trim or near-trim flights. Figure 2 shows the scenario that we will fly in this research.



Figure 2. Scene of urban application with unknown utility lines amid buildings. UAV must fly at low altitude to evade detection by enemy radar.

Figure 2 shows a cityscape with buildings and utility lines. The objective of the flight control system is to navigate to a known coordinate to a specific building. Utility lines and building constraints will be sensed and avoided during the flight, as they are “detected.” In the near term, we will “virtually sense” the obstacles since we will not have the sensor hardware available to us, and the guidance logic will react to that “detection.”

SUMMARY OF HYBRID MANEUVER LOGIC

The hybrid logic approach quantizes the systems dynamics into discrete maneuver and trim primitives. With this quantization, the approach reduces the computational complexity of the motion-planning problem for a nonlinear, high-dimensional dynamic vehicle. Specifically, we restrict the feasible nominal system trajectories into a family of time-parameterized curves that can be obtained by interconnecting the appropriately defined primitives. Instead of solving an optimal control problem over a high-dimensional, continuous space, we solve a mixed-integer-programming problem in a reduced-order space.

At the core of the control architecture lies a hybrid automaton whose states represent feasible trajectory primitives of the vehicle. The task of the automaton is to generate complete, feasible, and “optimal” trajectories by appropriately interconnecting the available primitives (the hybrid automaton’s states). In addition to reducing computational complexity, this approach provides a mathematical foundation for generating a provably stable hierarchical system to develop the tools to analyze robustness in the presence of model and environment uncertainty. Rigorous mathematical definitions of all these concepts are summarized in Ref. [1]. Figure 3 shows the key elements of the hybrid control architecture.

Equilibrium Points and Trim Trajectories

One class of a vehicle’s fundamental motions consists of its *trim trajectories*. Trim trajectories are those along which body velocities and the control input are a constant, or equivalently, trajectories along which a reduced set of the dynamics has no acceleration components. For aerial vehicles, trim trajectories are usually classified by steady-state values of speed (V), flight-path angle (γ) or angle-of-attack (α), heading turn rate (r), and sideslip (β) (see Ref. [3]). In our control architecture design, our first step involves selecting a number of feasible trim trajectories. We do this by gridding the set of attainable values of the steady-state parameters in the space of (V, γ, r, β). This set is compact but (discretely) spans the flight envelope of the aircraft. This step produces a set of trims $[S(V, \gamma, r, \beta)]$, which are computed off-line using the plant model. Since the trim library represents the vehicle dynamic response capabilities (in different trim conditions), it does not need to be recomputed unless the plant dynamic parameters change.

Trim states and trajectories are used extensively in high-complexity aerospace applications to simplify (linearize) vehicle dynamics and vehicle performance, and thus design closed-loop control gains as a function of a reduced parameter basis. Switching control systems are popular control system designs that switch through

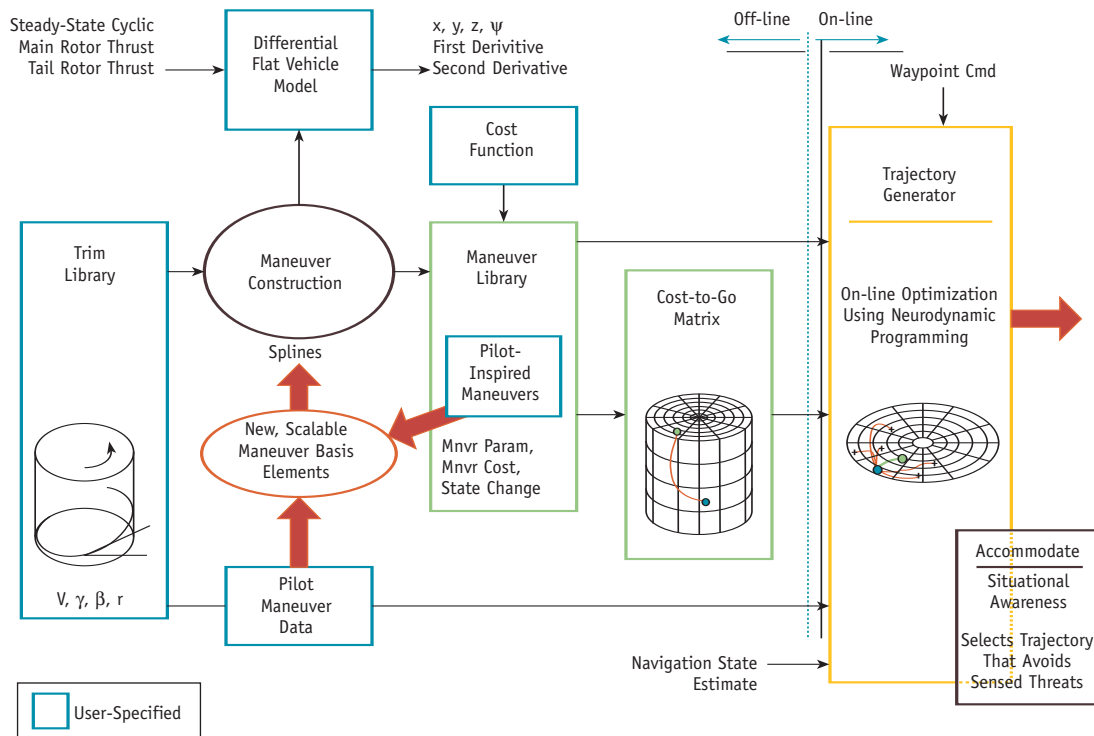


Figure 3. Key components of the hybrid logic automaton. During the off-line preprocessing phase, we construct trim and maneuver trajectories, and populate a cost-to-go matrix in the reduced dynamics space. During the on-line trajectory design phase, we solve a dynamic programming optimization to compute the transition time and index of the next maneuver to reach the goal in a time-optimal way.

a sequence of controllers to progressively take the system to a desired equilibrium operating condition. A smooth, parameter-dependent description of the system is said to have a linear parameter varying (LPV) form and essentially represents the same idea. Gain scheduled controllers are used on these LPV models, where the gains smoothly schedule on the same parameters used to define the model (see Ref. [4]).

The drawback of systems that are modeled and controlled around trim flight conditions is that the closed-loop system does not take advantage of the full operating envelope of the aircraft. It is limited to operating regimes that are near trim (low accelerations), the controlled systems transitions tend to be slow, and information about the true nonlinear systems' transient behavior when switching between operating regimes can produce undesirable behavior such as limit cycles.

Maneuver Primitives

Vehicle transitions away from trim conditions must be characterized properly in order to enable the flight control system to autonomously design and fly aggressively. We define a maneuver as a finite-time transition between any two trim trajectories introduced in the previous section. Note that a maneuver can include transitions between the same trim state: for example, maneuvers like barrel rolls and split-S enable rapid changes in flight direction in an inertial frame and are useful to have available.

We use a two-step method to synthesize the dynamically feasible maneuvers between trims. We first synthesize a trajectory in the reduced output space (i.e., in the space of (V, γ, r, β)) and then invert the dynamics to compute the associated trajectory in the input space of the vehicle in the manner of Ref. [5]. We introduce a high-order (7th order) spline function parameterized in time, between the trims s_A and s_B according to the following rule:

$$X_m(t) = \sum_{i=0}^7 \alpha_i * t^i, t \in [0, T_F] \quad (1)$$

such that

$$J(A \rightarrow B) = \min(T_F) \quad (2)$$

and

$$X_m(t = 0, \dots, t = T_F) \in \text{Allowable Space} \quad (3)$$

where

$$X_m(0) = s_A \quad \text{and} \quad X_m(T_F) = s_B \quad (4)$$

To ensure that the trajectories are dynamically realizable, we invert the input-output dynamic model and thus validate the trajectories in the input space; we had already ensured that the trajectories were realizable in the output space during the spline construction process. Like the trim library, maneuver primitives are also computed off-line from the vehicle model. Our reduced-order parameterization of the vehicle's dynamics is significant here, because it minimizes the order of the quantization space and therefore, the total amount of data that the motion planning problem can

use to completely derive its trajectories. Finally, as maneuver library elements are synthesized and stored, we also store the total spatial displacement effected by each motion element in 3D space, as well as the cost required to execute that maneuver.

Motion Planning

Given the trim and maneuver libraries introduced previously and assembled off-line, the on-line guidance logic selects the next maneuver (hybrid automaton state) from the vehicle's current trim flight condition and its transition time. The reachable maneuver space of any trim flight element is defined as the subset of all maneuvers that start at that trim state. The hybrid controller must compute the timing of the jump to the next maneuver in the reachable maneuver space that minimizes the cost function. Let X represent the trim state in the automaton and let h be a vector containing the position and heading of the vehicle. We want to drive the system to a desired trim state X^* , with position and heading H^* and define a running cost function $L(X, h)$ with $L(X^*, h^*) = 0$. Given a control policy μ , we define a total cost function.

$$J_\mu(X_0, h_0) = \int_0^\infty L(H, h) dt \quad (5)$$

$$J_\mu^*(X_0, h_0) = \min_{\mu^*} J_\mu \quad (6)$$

The guidance system computes the optimal control policy μ^* that minimizes total cost J in driving the system to the desired trim state. We use dynamic programming theory to solve this problem, specifically using Bellman's optimality equation to compute the optimal controls. The optimization solves a mixed integer programming problem with one continuous variable (the hybrid state transition time) and one discrete variable (the maneuver index of the next transition). Since the dimension of the state space has been reduced to one discrete variable and four continuous variables, neurodynamic programming approximation techniques for a compact representation of the cost function can be used effectively, making the control algorithms suitable for a real-time application.^{[6],[7]}

In our implementation, we solve a "motion planning" or "kinodynamic planning" problem whereby the trajectory planner incorporates system dynamics and nonholonomic constraints into its solutions. Furthermore, because the motion planner computes the optimal trajectory online by repeatedly solving an optimization problem for the optimal state (motion element) of the hybrid automaton, the planner embodies feedback control. Furthermore, the hybrid automaton encodes all the relevant dynamic information in the maneuver elements. As a result, the trajectory design process produces dynamically tractable trajectories that can include highly agile motions. Nevertheless, because we constrain all maneuvers to begin and terminate in a trim state, the resulting trajectory composed from the elements is necessarily stable.

Our guidance algorithm decomposes the motion planning problem into an off-line preprocessing part and an on-line part, and thus implements a dynamic

programming optimization in real time. In the off-line preprocessing phase, after the maneuver and trim trajectories are computed, we synthesize a table of costs-to-go. This is done by gridding the 3D space out to some known distance in each of the X, Y, and Z directions. The off-line planner computes the optimal trajectories and the associated costs-to-go from every grid point to every other grid point. A cost-to-go stores these computed optimal trajectory costs.

In the on-line phase of the problem, the optimizer computes the optimal maneuver strategy to apply next and its transition (initiation) time. Based on the predicted relative coordinates of the vehicle after it executes the maneuver (recall that the stored table contains data about which the relative displacements induced by each maneuver) the optimizer reads the cost-to-go from that maneuver termination point to the desired waypoint in the relative frame. The on-line optimization policy of Eq. (6) is thus implemented as

$$J_{\mu}^*(X_0, h_0) = S(X, h, \mu) + \int_{T_1}^{\infty} L'(X, h) dt \\ = S(X_0, h_0, \mu, X_{T_1}, h_{T_1}) + J^*(X_{T_1}, h_{T_1}) \quad (7)$$

where $J^*(X_{T_1}, h_{T_1})$ is the optimal cost-to-go from the stored table of costs and $S(X, h)$ is the cost of the particular hybrid logic state (for the T_1 seconds until the end of that maneuver). Thus, the optimizer returns the state index that the hybrid logic automaton must next occupy and its transition time, which minimizes the infinite horizon optimal control problem and the feedforward control trajectory necessary to implement it. Thus, the dynamic programming approach enables us to compute the optimal control policy without having to explicitly solve for the entire trajectory at every instance in time.

OBSTACLE AVOIDANCE

In our final application, the vehicle must fly between buildings that it may know about *a priori* and avoid wires that we will not know about until we are within 20 m and in line of sight of them. The onboard dynamic programming subsystem of the hybrid logic typically computes the index of the next maneuver to execute and its switching time. We have made some simple modifications in the hybrid logic algorithm to preclude collisions in the feedback trajectory design.

The off-line part of the analysis will handle any known obstacles by synthesizing and costing those trajectories that avoid the obstacles when it computes the cost-to-go matrices. As in the nominal algorithm, the analysis will score (compute the cost of) each maneuver and classify the relative state change between the initial and final coordinates induced by the maneuver. However, the cost-to-go matrix will compute optimal costs to go from one cell to another in the physical environment that do not intersect any obstacles. This will enable us to encode the known obstacles in the cluttered terrain in the cost map.

Based on the cost function and the desired goal state as well as the maneuver cost and cost-to-go matrix, the

hybrid logic algorithm identifies the maneuver that promotes a state change and its cost, such that the sum of that cost and the cost-to-go from the new state to the desired goal state is the minimum achievable of all feasible trajectories covered by the graph. To handle collision avoidance, we will include a verification step into this dynamic programming principle. Whereas the canonical implementation merely uses the net state change induced by the maneuver, the collision avoidance logic will simulate the candidate trajectory element at discrete points in time through the completion of the trim phase and the end of the identified maneuver element to check if this time-optimal trajectory collides with the wires. If so, the candidate maneuver element is discarded, another candidate is chosen from the reachable set of maneuvers, and simulated for collision checking. Note that the logic only simulates the known segment of the trajectory that was computed in a finite time interval, and does not necessarily attempt to solve for the complete trajectory to the next waypoint at that time. The process repeats until the planner finds the (sub)optimal solution that does not violate the collision constraint.

DESCRIPTION OF THE EXPERIMENTS

In our final experiments as part of the SEC program, we will simulate the wire and building detection sensors in our guidance logic. The cable information will not be known during the cost-map construction phase, however, building locations will be known and will be reflected in the costs. The real-time, onboard part of the guidance algorithm will respond to incoming information about unknown obstacle locations during the flight, since it will validate the trajectory associated with the “optimal” maneuver selection logic to find a feasible path. We are integrating our simulation in the GTMAX framework.

In our integration, algorithm test, and implementation experiments toward the Phase II flight test demonstrations so far, we have integrated our guidance logic with the GTMAX flight control system software and have defined a racetrack scenario that the vehicle must traverse. The GTMAX flight software interface to the guidance module requires that we qualify the mission (i.e., the racetrack) via a series of waypoints, provide the associated (nominal) vehicle speed and heading with which the vehicle must approach the waypoint. The agile, hybrid logic software synthesizes the trajectory commands to the vehicle flight controller, and the vehicle aggressively flies between the waypoints, approaching, maneuvering around, and leaving the waypoint at the defined speeds in agile, optimal (minimum time), dynamically feasible trajectories.

We have integrated our hybrid logic-based agile, feedback guidance algorithm with the Georgia Tech flight control system. Figures 4 and 5 show two images of the mission scene consisting of eight waypoints laid out in the race-track; the yellow trace shows the nearly completed mission trajectory. These images also show the views the operator sees via the scene generator capabilities of the flight simulator.

Figure 4 shows the mission scenario. The helicopter lifts vertically off from coordinates $(0, 0, 0)$ at left bottom of the image and flies vertically to a heading hold position at $(0, 0, -30 \text{ ft})$. From this point, the Draper guidance logic activates and flies the vehicle nearly all the way around the racetrack. Figure 5 shows the resulting trajectory tracks without the scene generator embellishments such as waypoint markings.

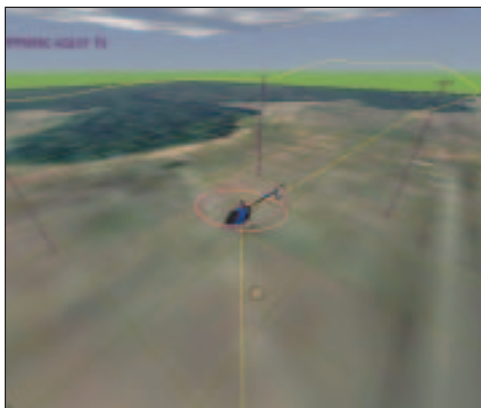


Figure 4. Illustration of mission scenario. The helicopter lifts off from the left bottom corner of the image. It completes the racetrack in a clockwise direction, returning to the initial takeoff coordinates $(0, 0, -30 \text{ ft})$.

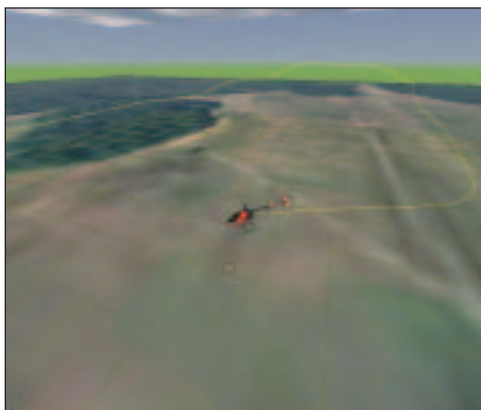
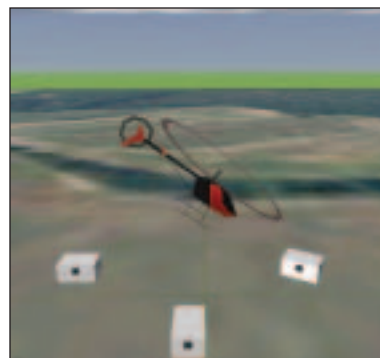


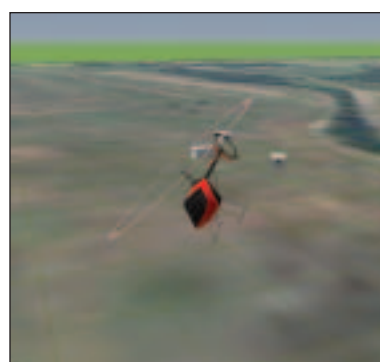
Figure 5. Different perspective view of the flight traces without the waypoint markings.

In Figure 4, the vertical lines with circles atop them mark the waypoints. For completeness, we have shown the true scene of the same flight with the flight trace but without marking the waypoints in Figure 5.

Figure 6 presents some views of agile elements during the flight as the vehicle accelerates and decelerates, or ascends and descends between the waypoints. The onboard dynamic programming engine of our optimal guidance logic autonomously generated these maneuvers to minimize time of flight between the waypoints.



Pitch down to accelerate forward



Banked turn at 30 ft/s to induce 270-deg turn



Flare to decelerate

Figure 6. Three agile, autonomously-generated flight elements.

SUMMARY

In this paper, we present a hybrid automaton-based agile trajectory design method with collision avoidance constraints to enable autonomous missions in an urban terrain with unknown threats. We have simulated the algorithm in 3D space without the constraint checking, and are currently implementing the algorithm on an Excel helicopter. We set up and solved a dynamic programming problem so that the solution consists of

optimally sequencing maneuver primitives from among a set of discrete maneuvers that quantize the vehicle's dynamics into (dynamically) feasible trims and agile maneuvers states. A hybrid automaton generates feasible state (maneuver) transitions between the vehicle's current trajectory state and a reachable trajectory such that an interconnection of such transitions will minimize the cost to reach the next waypoint. We further simplify the task of the online hybrid automaton by setting up a dynamic programming problem; we discretize the reachable physical space of the vehicle and compute the optimal costs-to-go from any one point to another point in this discretized physical map. This approach reduces optimization complexity for real-time implementability, and involves merely computing the best next maneuver state to transition to and accessing the cost-to-go matrix to find the total cost of the optimal trajectory. Thus, the effort of computing the optimal solution is divided into a computationally-intensive cost-to-go computation that happens off-line and a time-efficient finite-horizon search on-line.

We are also adapting the real-time algorithm to handle threats such as utility lines in the urban cityscape that are only known after they are detected by onboard sensors. We do this by simulating the candidate maneuver or trim indices within some finite horizon to check for collisions; if a collision is predicted, we reject that candidate solution and try the next.

REFERENCES

- [1] Frazzoli, E., M.A. Dahleh, and E. Feron, "A Hybrid Control Architecture for Aggressive Maneuvering of Autonomous Helicopters," IEEE Conf. on Decision and Control, December 1999.
- [2] McConley, M.W., M.D. Piedmonte, B.D. Appleby, E. Frazzoli, E. Feron, and M.A. Dahleh, "Hybrid Control for Aggressive Maneuvering of Autonomous Aerial Vehicles," 19th Digital Avionics Systems Conference, October 2000.
- [3] Stevens, B. and F. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, 1992.
- [4] Shamma, J. and J. Cloutier, "Gain Scheduled Missile Autopilot Design Using Linear Parameter Varying Transformations," *AIAA Journal of Guidance, Control, and Dynamics*, March 1993, pp. 256–263.
- [5] Petit, N., M. Milam, and R. Murray, "Inversion Based Trajectory Optimization," IFAC Symposium on Nonlinear Control Systems Design, 2001.
- [6] Bertsekas, D. and J. Tsitsiklis, *Neurodynamic Programming*, Athena Scientific, 1996.
- [7] Bertsekas, D., *Dynamic Programming and Optimal Control*, Athena Scientific, 1995.



BIOGRAPHIES

LEENA SINGH is a Senior Member of the Technical Staff in the Autonomous Control Group. She has 7 years of research experience exploring new autonomous guidance, navigation, and control algorithms for systems, which have included UAVs, helicopter fire control systems, and aircraft engines. Her recent research focuses on envelope extending advanced autonomous G&C algorithms. Dr. Singh received PhD and MS degrees from Rensselaer Polytechnic Institute (RPI) in 1997 and 1993, respectively, and a BS in Physics and Mathematics from Mt. Holyoke College.

JOHN PLUMP is a Principal Member of the Technical Staff in the Autonomous Control Group at Draper. He has 18 years of experience in the development of guidance and control algorithms and embedded software for autonomous underwater and air vehicles. His current work concerns guidance and control strategies for small, unpowered parafoils. Mr. Plump received MS and BS degrees in Mechanical Engineering from MIT in 1986 and 1983, respectively.

MARC MCCONLEY is a Principal Member of the Technical Staff in the Autonomous Control Group at Draper. He has 7 years of experience in the development of guidance, navigation, and control algorithms and software for autonomous vehicles and guided gun-fired munitions. His current work is focused on advanced navigation and mapping technology development and Deep Integration for INS/GPS navigation. Dr. McConley received PhD and SM degrees in Aeronautics and Astronautics from MIT in 1997 and 1994, respectively, and a BSE degree in Mechanical and Aerospace Engineering from Princeton University in 1992.

BRENT APPLEBY is the Autonomous Control Group Leader. He has worked on numerous guidance, navigation, and control projects for a variety of applications, including autonomous rotorcraft, parafoils, undersea vehicles, guided projectiles, and a gun-launched UAV, as well as flexible space structures, satellites, and the Space Shuttle autopilot. He is also a Lecturer in the Aeronautics and Astronautics Department at MIT. Dr. Appleby received SM, SM, and PhD degrees from MIT.