Implementation and Flight Test Results of MILP-based UAV Guidance

Tom Schouwenaars* Mario Valenti* Eric Feron* Jonathan How**
*Laboratory of Information and Decision Systems
**Aerospace Controls Laboratory
Massachusetts Institute of Technology
77 Massachusetts Ave, Room 32D-712
Cambridge, MA 02139, USA
617-253-7446
{toms,valenti,feron,jhow}@mit.edu

Abstract—This paper discusses the implementation of a guidance system based on Mixed Integer Linear Programming (MILP) on a modified, autonomous T-33 aircraft equipped with Boeing's UCAV avionics package. A receding horizon MILP formulation is presented for safe, real-time trajectory generation in a partially-known, cluttered environment. Safety at all times is guaranteed by constraining the intermediate trajectories to terminate in a loiter pattern that does not intersect with any no-fly zones and can always be used as a safe backup plan. Details about the real-time software implementation using CPLEX and Boeing's OCP platform are given. A test scenario developed for the DARPA-sponsored Software Enabled Control Capstone Demonstration is outlined, and simulation and flight test results are presented.

TABLE OF CONTENTS

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) have been used by both military and civilian organizations in a number of applications. Recent advances in guidance technologies have enabled some UAVs to execute simple mission tasks without human interaction. Many of these tasks are pre-planned using reconnaissance or environment information. For example, air operations are executed according to an Air Tasking Order (ATO), which may take up to 72 hours to plan, task and execute [15]. In volatile situations, however, information about the vehicle's operating environment may be limited: a detailed map of the environment might not be available ahead of time, and obstacles might be detected while a mission is carried out. In such situations, task planning flexibility and safe trajectory solutions are essential to the survivability and success of the autonomous system. Most unmanned vehicles, however, do not exhibit this level of performance. Intelligent guidance systems providing the required flexibility are therefore a topic of active research.

An approach to adaptive autonomous trajectory planning based on Mixed Integer Linear Programming (MILP) was recently introduced in [10]. MILP is a powerful mathematical programming framework that extends continuous linear programming to include binary or integer decision variables [5]. These variables can be used to model logical constraints such as obstacle and collision avoidance rules, while the dynamic and kinematic properties of the vehicle are formulated as continuous constraints. Thanks to the increase in computer speed and implementation of powerful state-of-the-art algorithms in software packages such as CPLEX [1], MILP has become a feasible option for real-time path planning, as demonstrated by the results discussed in this paper.

Under the DARPA-sponsored Software Enabled Control (SEC) program, a receding horizon MILP formulation was developed for safe, real-time trajectory generation in a partially-known, cluttered environment [11]. After each time interval of a certain duration, a new MILP problem is solved that incorporates updated information about the environment, the task and the state of the vehicle, and is constrained to terminate in a safe loiter pattern. The output of the MILP optimization is a sequence of waypoints constituting a partial trajectory to the goal. As such, an optimal reference trajectory achieving a particular task, such as to search for a target in a partially unknown area, is computed in real-time, i.e. as the mission unfolds.

This paper discusses the implementation of this MILP-based guidance algorithm on a modified, autonomous T-33 aircraft, augmented with Boeing's UCAV avionics package. The planning software using CPLEX's Concert Technologies [1] was integrated with Boeing's real-time Open Control Platform

0-7803-8870-4/05/$20.00 ©2005 IEEE
IEEEAC paper # 1396

(OCP) [7] and flight tested during the capstone demonstration of the SEC program in June 2004. A scenario was flown in which the T-33 acted as a UAV that was given mission level commands by an F-15 weapon systems officer (WSO). The communication between the F-15 WSO and the T-33 UAV was done using a natural language interface, that interpreted human language commands of the WSO and transformed them in real-time into input data for the optimal guidance problem. These flight tests marked the first time an on-board Mixed Integer Linear Programming-based guidance system was used to control a UAV.

The paper is organized as follows. Section 2 gives an overview of the SEC experiment and the associated technology development. Section 3 describes the basic trajectory planning problem and our approach to maintaining safety. The detailed MILP formulation of the latter is presented in Section 4. Next, Section 5 discusses the real-time software implementation and some practical engineering decisions. Simulation and flight test results are presented in Section 6, and Section 7 concludes.

Figure 1. Boeing F-15E Strike Eagle

Figure 2. Lockheed T-33

2. EXPERIMENT OVERVIEW

Mission Scenario

As originally discussed in [6], as part of the DARPA-sponsored Software Enabled Control Program, our team was tasked with developing a flight-test scenario that exhibited UAV technology developed at MIT. For this demonstration, we had access to two flight assets: a Boeing F-15E fighter jet (similar to the aircraft shown in Figure 1) and a Lockheed T-33 trainer fighter jet (similar to the aircraft shown in Figure 2), equipped with Boeing's UCAV avionics package. The former was to be flown by a pilot and will be referred to as the Fixed-Wing (FW) vehicle. The latter was to be guided by our technology and will be referred to as the Unmanned Aerial Vehicle (UAV). Besides these aircraft, a ground station receiving state and user-defined information from both vehicles was available to monitor the experiment.

To enable a hard real-time execution, our demonstration software needed to be integrated with Boeing's Open Control Platform (OCP) [7] and loaded onto a laptop fitted in each aircraft. The OCP software provided us with an aircraft interface including the following abilities:

- Send and receive state and user-defined data between both aircraft using a Link-16 communications interface,
- Receive the current vehicle state data,
- Send a set of pre-defined commands to the aircraft avionics system which include Set and Hold Turn Rate, Set and Hold Speed, Set and Hold Altitude, Set and Hold Heading, and
- Memory storage and time frame execution.

Given these demonstration resources, we developed a mission scenario (shown in Figure 3) in which the UAV performs tasks in support of the FW vehicle:

Mission— A manned fighter aircraft (FW) and a UAV will work together on a mission to collect images of a possible site in enemy territory. The FW Weapon Systems Officer (WSO) will communicate with the UAV using a Natural Language Interface, which allows the FW WSO to speak with the UAV using normal sentence commands. The UAV will perform the reconnaissance for the mission in a partially-known environment, and the FW WSO will decide how the UAV will be used to accomplish the mission goals. The UAV will possess the ability to detect threats and collect images, whereas, if applicable, the FW vehicle will be able to deliver weapons. Since the environment is only partially-known, there may be threats to both the manned and unmanned aircraft.

Starting Condition—The UAV will start in a pre-defined loiter pattern; the FW vehicle will be flying an air-patrol near the enemy territory. The environment is partially-known and updated in real-time to both the UAV and the FW. A pop-up threat may arise en route to the search site, which is currently unknown.

Mission Narrative—

1: The FW vehicle is commanded to gather information and possibly destroy an enemy site located in unknown territory. Because of the mission risk, the FW vehicle assigns the UAV, stored in a nearby airspace volume, to gather information at the designated site. The UAV leaves the loiter area and moves

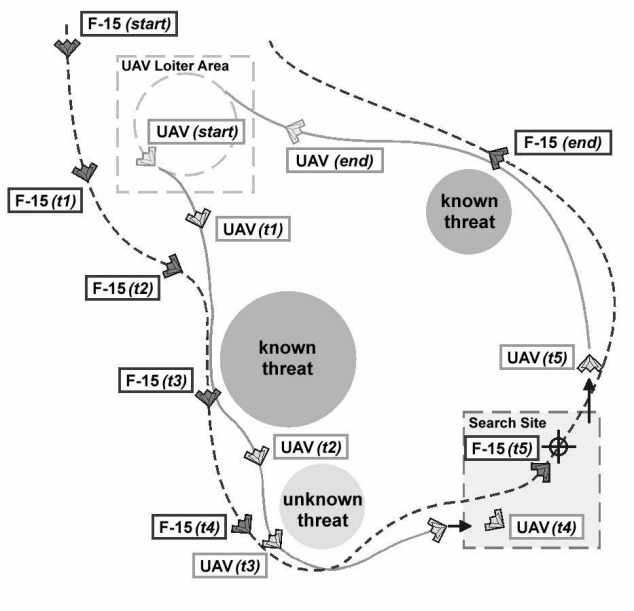

Figure 3. MIT Flight Experiment Mission Overview

toward the designated task area. The F-15 follows behind at a higher altitude and a safe distance.

2: The UAV is informed of a pop-up threat en route to the task area. The UAV accounts for the threat dynamically, automatically generates a revised safe trajectory around the threat and other no-fly zones, while notifying the FW vehicle of the threat's position.

3: As the UAV moves within a few minutes of the task location, the UAV notifies the FW vehicle of its location. At this point, the FW vehicle will provide the UAV with the exact ingress and egress conditions into and out of the search area. The UAV modifies its flight path to arrive at the site as commanded.

4: The UAV enters the site, notifies the FW of its location and begins its search for the target.

5: The UAV identifies the target and sends an image to the FW for evaluation. The FW commands the UAV to return to its original loiter area, while the FW prosecutes the target.

Exit Conditions—The UAV safely returns to the original predefined loiter pattern location; the FW vehicle returns to flying an air-patrol near the enemy territory.

Technology Development

The above mission scenario allowed us to demonstrate technology developments in several areas, leading to three distinct software components:

1: *Natural Language Interface* — This component interprets and converts normal sentence commands (e.g., "Search this region for threats") from the humans on-board the FW vehicle into data the UAV can use and understand and vice-versa. It is aimed at minimizing the workload of the WSO when interacting with the computer-based UAV. Namely, the WSO can give the UAV high level mission commands in English rather than low level guidance commands such as "Turn left" or "Speed up".

2: *Task Scheduling and Communications Interface* — The primary goal of this component is to interpret the command data from the Natural Language Interface and develop a series of tasks the vehicle can perform. The list of mission tasks that were developed includes: flying to a waypoint X, entering a loiter pattern, and performing a search pattern. For each of these tasks, the user must provide the task ingress/egress conditions, and the size and location of the task area. The component also contains the communications processing module that provides the FW WSO with the authority to send task commands and receive status updates, threat/obstacle avoidance information, and acknowledgement messages. In addition, it provides the FW WSO with the ability to override the UAV's navigation system in the event of an emergency or error.

3: *Trajectory Generation Using MILP* — After the Natural Language Interface and Task Scheduling component have converted the mission steps into a series of tasks for the vehicle to perform, the Trajectory Generation Module guides the vehicle from one task location to the next. Time-optimal safe trajectories that account for the current state of the vehicle and the knowledge of the environment are computed using Mixed Integer Linear Programming (MILP). Since we assume that in the mission scenario the environment is only partially-known and explored in real-time, the MILP guidance algorithm uses a receding horizon planning strategy, allowing for online trajectory computation.

In this paper, we will focus on the MILP Trajectory Generation component. Further details about the Natural Language Interface can be found in [4] and [9] ; a description of the Task Scheduling and Communications Interface and details about the integrated mission system are given in [14].

3. SAFE TRAJECTORY PLANNING

Basic Planning Problem

The basic problem tackled by the Trajectory Generation module is to guide the UAV from an initial state to a desired one through an obstacle field while optimizing a certain objective. The latter can be to minimize time, fuel or a more sophisticated cost criterion such as to minimize visibility or to maximize the total area explored. We consider 2D scenarios in which no-fly zones or "obstacles" are detected while the mission is carried out, but assume that the environment is fully characterized inside a certain detection region $\mathcal{D}$ around the aircraft. For the SEC demonstration, we considered a circular region of radius 9 mi and assumed that all obstacles $\mathcal{O}$ within that radius were static. The MILP formulation that we will present, however, can readily be generalized to account for any detection shape, such as a radar cone, and for unknown

areas within that shape.

Since trajectories must be dynamically feasible, the vehicle dynamics and kinematics should be accounted for in the planning problem. For optimization purposes, the vehicle is characterized by a discrete time, linear state space model (**A**,**B**) in an inertial 2D coordinate frame. As such, the state vector $\mathbf{x}$ consists of the position (x, y) and inertial velocity $(\dot{x}, \dot{y})$. Depending on the particular model, the input vector $\mathbf{u}$ is an inertial acceleration or reference velocity vector. In both cases, however, together with additional linear inequalities in $\mathbf{x}$ and $\mathbf{u}$, the state space model must capture the closed-loop dynamics that result from augmenting the vehicle with a waypoint tracking controller.

Since the environment is only partially-known and explored in real-time, a receding horizon planning strategy is used to guide the vehicle towards the desired destination. The latter is denoted by $\mathbf{x}_f$ and is an ingress/egress state of a task or some other waypoint with a corresponding inertial velocity vector. At each time step, a partial trajectory from the current state towards the goal is computed by solving the trajectory optimization problem over a limited horizon of length T. Because of the computation delay, the initial state $\mathbf{x}_0 = (x_0, y_0, \dot{x}_0, \dot{y}_0)$ in the optimization problem should be an estimate $\mathbf{x}_{estim}$ of the position and inertial velocity of the aircraft when the plan is actually implemented.

The solution to the optimization problem provides a sequence of waypoints (x_i, y_i) and corresponding inertial reference velocities $(\dot{x}_i, \dot{y}_i)$ to the aircraft for the next T time steps. Typically, however, only the first waypoint and reference velocity of this sequence are given to the waypoint follower, and the process is repeated at the next time step. As such, new information about the state of the vehicle and the environment can be taken into account at each time step.

By introducing a cost function J_T over the T time steps, the general trajectory optimization problem can be formulated as follows:

$$\min_{\mathbf{x}_i, \mathbf{u}_i} J_T = \sum_{i=0}^{T-1} \ell_i(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_f) + f(\mathbf{x}_T, \mathbf{x}_f) \tag{1}$$

subject to:

$$\begin{cases} \mathbf{x}_{i+1} &= \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{u}_i,\ i = 0 \dots T-1 \\ \mathbf{x}_0 &= \mathbf{x}_{estim} \\ \mathbf{x}_i &\in \mathcal{X}_0 \\ \mathbf{u}_i &\in \mathcal{U}_0 \\ (x_i, y_i) &\in \mathcal{D}_0 \\ (x_i, y_i) &\notin \mathcal{O}_0 \end{cases} \tag{2}$$

The objective function (1) contains a terminal cost term $f(\mathbf{x}_T, \mathbf{x}_f)$, which accounts for an estimate of the cost-to-go from the last state $\mathbf{x}_T$ in the planning horizon to the goal state $\mathbf{x}_f$. The sets $\mathcal{X}_0$ and $\mathcal{U}_0$ represent the (possibly non-convex) linear constraints on the vehicle dynamics and kinematics, such as bounds on velocity, acceleration and turn rate. Here, the 0-subscript denotes the fact that these constraints can be dependent on the initial state. Lastly, the expressions $(x_i, y_i) \in \mathcal{D}_0$ and $(x_i, y_i) \notin \mathcal{O}_0$ capture the requirement that the planned trajectory points should lie *inside* the known region $\mathcal{D}_0$, but *outside* the obstacles $\mathcal{O}_0$ as given at the current time step $i = 0$.

Loiter Principle

Despite these constraints, however, the above receding horizon strategy has no safety guarantees regarding avoidance of obstacles in the future. Namely, the algorithm may fail to provide a solution in future time steps due to obstacles that are located beyond the surveillance and planning radius of the vehicle. For instance, when the planning horizon is too short and the maximum turn rate relatively small, the aircraft might approach a no-fly zone too closely before accounting for it in the trajectory planning problem. As a result, it might not be able to turn away in time, which translates into the optimization problem becoming infeasible at a certain time step.

In Ref. [11], we therefore proposed a *safe* receding horizon scheme based on maintaining a known feasible trajectory from the final state $\mathbf{x}_T$ in the current planning horizon towards an obstacle-free loiter pattern. The latter must lie in the region of the environment that is fully characterized at the current time step and is computed and updated online. Assuming that the planned trajectories can be accurately tracked, at each time step, the remaining part of the previous plan together with the loiter pattern can then always serve as a safe backup or "rescue" plan. Namely, if at a certain time step, the guidance software fails to find a feasible solution to the optimization problem within the timing constraints of the real-time system, the aircraft can just keep following the previous trajectory. Eventually, that trajectory will end in a loiter pattern in which the aircraft can safely remain for an indefinite period of time, *i.e.*, without flying into an obstacle or no-fly zone. At each time step, a *safe* trajectory that is consistent with the known environment can thus be determined.

More specifically, for the SEC demonstration, we explicitly constrained the final state $\mathbf{x}_T$ to be an ingress state to a right *or* left turning loiter circle, respectively denoted by $\mathcal{C}_R$ and $\mathcal{C}_L$. As discussed in Ref. [11] and detailed in Section 4, this can be done by expressing N sample points along the loiter circles as affine functions of $\mathbf{x}_T$, thereby accounting for the fact that the minimum turn radius scales inversely with the velocity. This scaling property and the choice of turning direction give the aircraft more degrees of freedom to fit the loiter circles in the obstacle-free areas of the known environment. Introducing an index j for the sample points, the above safety constraint can be specified as follows:

$$\begin{cases} \mathcal{C}_R(\mathbf{x}_T) = \{(x_{Rj}, y_{Rj})\} &\in \mathcal{D}_0,\ j = 1 \dots N \\ \mathcal{C}_R(\mathbf{x}_T) = \{(x_{Rj}, y_{Rj})\} &\notin \mathcal{O}_0 \\ \text{OR} & \\ \mathcal{C}_L(\mathbf{x}_T) = \{(x_{Lj}, y_{Lj})\} &\in \mathcal{D}_0,\ j = 1 \dots N \\ \mathcal{C}_L(\mathbf{x}_T) = \{(x_{Lj}, y_{Lj})\} &\notin \mathcal{O}_0 \end{cases} \tag{3}$$

4. MILP FORMULATION

Mixed Integer Linear Programming

The optimal safe trajectory planning problem outlined above lends itself well to be formulated as a *mixed integer linear program* (MILP). MILP is a powerful mathematical programming framework that allows inclusion of integer variables and discrete logic in a continuous linear optimization problem [5]. It is commonly used in Operations Research [13], and has more recently been introduced to the field of hybrid systems [3] and trajectory optimization [10]. In our case, the continuous optimization is done over the states and inputs; the discrete logic is introduced by nonconvex constraints such as obstacle avoidance and minimum speed requirements, and by the binary selection between the right and left turning loiter circles.

As an illustration of how logical decisions can be incorporated in an optimization problem, consider the following abstract example. Assume that a cost function $J(\mathbf{x})$ needs to be minimized subject to either one of two constraints $\ell_1(\mathbf{x})$ and $\ell_2(\mathbf{x})$ on the continuous decision vector $\mathbf{x}$:

$$\begin{array}{l} \min\limits_{\mathbf{x}} J(\mathbf{x}) \\ \text{subject to:} \\ \qquad \ell_1(\mathbf{x}) \leq 0 \\ \text{OR} \quad \ell_2(\mathbf{x}) \leq 0 \end{array} \tag{4}$$

By introducing a large, positive number M and a binary variable b, this optimization problem can equivalently be formulated as follows:

$$\begin{array}{l} \min\limits_{\mathbf{x}} J(\mathbf{x}) \\ \text{subject to:} \\ \qquad \ell_1(\mathbf{x}) \leq Mb \\ \text{AND} \quad \ell_2(\mathbf{x}) \leq M(1-b) \\ \qquad\quad b \in \{0,1\} \end{array} \tag{5}$$

When $b = 0$, constraint $\ell_1(\mathbf{x})$ must be satisfied, whereas $\ell_2(\mathbf{x})$ is relaxed. Namely, if M is chosen sufficiently large, inequality $\ell_2(\mathbf{x}) \leq M(1-b)$ is always satisfied, independently of the value of $\mathbf{x}$. The situation is reversed when $b = 1$. Since b can only take the binary values 0 or 1, at least one of the constraints $\ell_1(\mathbf{x})$ and $\ell_2(\mathbf{x})$ will be satisfied, which is equivalent to the original "OR"-formulation in problem (4). In the special case where $J(\mathbf{x})$, $\ell_1(\mathbf{x})$ and $\ell_2(\mathbf{x})$ are (affine) linear expressions, problem (5) is a MILP.

The formulation can easily be extended to account for multiple constraints $\ell_k(\mathbf{x}),\ k = 1 \ldots K$, out of which at least N must be satisfied simultaneously. This is done as follows:

$$\begin{array}{l} \min\limits_{\mathbf{x}} J(\mathbf{x}) \\ \text{subject to:} \\ \qquad \ell_k(\mathbf{x}) \leq Mb_k,\ k = 1 \ldots K \\ \qquad \sum_k b_k \leq K - N \\ \qquad\quad b_k \in \{0,1\} \end{array} \tag{6}$$

The additional summation constraint ensures that at least N of the binary variables b_k are 0, thus guaranteeing that at least N of the inequalities $\ell_k(\mathbf{x}) \leq 0$ are satisfied simultaneously.

MILP Trajectory Planning Formulation

We now apply the above framework to the trajectory optimization problem of Section 3.

State Space Model—System identification experiments using the UAV DemoSim simulation software provided by Boeing gave us insight into the velocity response of the UAV. We decided that for guidance purposes a piece-wise linear first order approximation with an actuator delay was sufficient. We identified the time constant and DC gain of the transfer function for a discrete set of forward velocities (from 350 fps to 500 fps with a resolution of 10 fps) and stored these in a look-up table. At each iteration of the receding horizon strategy, the model corresponding to the velocity at that time step was used, thus linearizing the nonlinear response into several LTI modes scheduled around the initial velocity.

Taking the desired inertial velocity as input, we used the following continuous state space model:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{\tau_l} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_l} \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{k_l}{\tau_l} & 0 \\ 0 & \frac{k_l}{\tau_l} \end{bmatrix} \begin{bmatrix} \dot{x}_{cmd}(t) \\ \dot{y}_{cmd}(t) \end{bmatrix} \tag{7}$$

Here, τ_l is the time constant and k_l is the gain corresponding to the l^{th} LTI mode. Note, however, that these dynamics are homogeneous in the x- and y-coordinate and as such ignore differences in the lateral and longitudinal aircraft dynamics. As will be detailed further on, we introduced additional constraints on the state and input vectors to correct for this.

Since the typical time constant was around 9.7s, using the bilinear transform, we discretized the above model with a time step of $\Delta t = 10$ s, thus obtaining:

$$\begin{aligned} &[x_{i+1}\ y_{i+1}\ \dot{x}_{i+1}\ \dot{y}_{i+1}]^T = \\ &\begin{bmatrix} 1 & 0 & \frac{\Delta t}{2}\left(1 + \frac{2\tau_l - \Delta t}{2\tau_l + \Delta t}\right) & 0 \\ 0 & 1 & 0 & \frac{\Delta t}{2}\left(1 + \frac{2\tau_l - \Delta t}{2\tau_l + \Delta t}\right) \\ 0 & 0 & \frac{2\tau_l - \Delta t}{2\tau_l + \Delta t} & 0 \\ 0 & 0 & 0 & \frac{2\tau_l - \Delta t}{2\tau_l + \Delta t} \end{bmatrix} \\ &\begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} + \begin{bmatrix} \frac{k_l(\Delta t)^2}{2\tau_l + \Delta t} & 0 \\ 0 & \frac{k_l(\Delta t)^2}{2\tau_l + \Delta t} \\ \frac{2k_l \Delta t}{2\tau_l + \Delta t} & 0 \\ 0 & \frac{2k_l \Delta t}{2\tau_l + \Delta t} \end{bmatrix} \begin{bmatrix} \dot{x}_{cmd,i} \\ \dot{y}_{cmd,i} \end{bmatrix} \end{aligned} \tag{8}$$

Together with the extra constraints, we found this model to

produce good results for tasks requiring intensive waypoint tracking and sharp turns (i.e. the loiter and search tasks). However, for straight trajectories with more or less constant speed, we also found it to react too aggressively to perturbations along the nominal trajectory. To transition between two task areas, we therefore used a more conservative double integrator model that does not distinguish between the different LTI modes:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \dot{x}_{i+1} \\ \dot{y}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_i \\ \ddot{y}_i \end{bmatrix}. \quad (9)$$

where we again used $\Delta t = 10$ s.

Initial State Estimate— Since it takes a certain time to compute the trajectory, the initial state should be an estimate of what the vehicle's state will be when the plan is actually implemented. In our case, the computation delay was 1 s. In addition, we accounted for a 1.2 s actuator delay. To obtain an estimate of the initial state at the next iteration, we thus propagated the dynamics following the previous plan for 2.2 s.

Cost Function—Given the above models, we were interested in guiding the UAV between waypoints in the fastest possible way, while avoiding no-fly zones. The exact shortest time between two states, however, can only be computed if the planning horizon spans that arrival time, or if an exact cost-to-go is known. Since in the scenario of interest the environment is not characterized beyond a certain detection radius around the vehicle, computing an exact time-to-go function as proposed in [2] is not possible. We therefore opted for the following heuristic approach.

In case there are no known obstacles intersecting the straight line between the waypoint and the current location, that waypoint is used as the desired state in the cost function. In case there are obstacles blocking this direct line-of-sight, the shortest path (as far as the known obstacles are concerned) must go through one of the visible corner points of these no-fly zones. This point can then act as an intermediate waypoint en route to the final destination. To determine this optimal intermediate point, a grid is constructed between the corner points of all known obstacles interfering with the line of sight. A shortest path algorithm is then run to compute the approximate shortest time towards the goal from each visible corner point, thereby assuming the UAV is flying at maximum speed. As such, the "best" intermediate waypoint is determined by minimizing the total time from the current location to one of the visible vertices and from that point to the destination as given by the approximate cost-to-go function.

Using this intermediate (or, in the obstacle-free case, the original) waypoint $\mathbf{p}_f = (x_f, y_f)$, we used the following piecewise linear cost function that aims at designing a *fast* trajectory between the initial position $\mathbf{p}_0 = \mathbf{p}_{estim} = (x_0, y_0)$ in the planning horizon and $\mathbf{p}_f$:

$$\min J = \sum_{i=0}^{T} -q\mathbf{v}_i'(\mathbf{p}_f - \mathbf{p}_{estim}) + r|\mathbf{p}_i - \mathbf{p}_f| \quad (10)$$

The first term tries to maximize the scalar product of the inertial velocity $\mathbf{v}_i = (\dot{x}_i, \dot{y}_i)$ with the vector that is pointing from the initial position to the desired one. The effect is twofold: it will speed the aircraft up to its maximal velocity, while turning it toward the right direction. In addition, the term $r|\mathbf{p}_i - \mathbf{p}_f|$ tries to minimize the 1-norm distance towards the goal. Both terms thus work towards the same objective, with q and r weighting the two contributions.

If, at a certain iteration, the planned trajectory passes through or near waypoint $\mathbf{p}_f$ at a time step $T_f \leq T$ in the planning horizon, the cost function for the next iteration is split into two parts:

$$\begin{aligned} \min \tilde{J} &= \sum_{i=0}^{T_f-1} -q_f\mathbf{v}_i'(\mathbf{p}_f - \mathbf{p}_{estim}) + r_f|\mathbf{p}_i - \mathbf{p}_f| \\ &+ \sum_{i=T_f}^{T} -q_n\mathbf{v}_i'(\mathbf{p}_n - \mathbf{p}_f) + r_n|\mathbf{p}_i - \mathbf{p}_n| \quad (11) \end{aligned}$$

in which $\mathbf{p}_n$ is the next (intermediate) waypoint. The first $T_f - 1$ time steps are thus used to minimize the cost towards waypoint $\mathbf{p}_f$; the remaining steps aim at minimizing the cost towards the next waypoint $\mathbf{p}_n$. As a result, depending on the relative weighting, the MILP will produce a trajectory that will pass through or close by $\mathbf{p}_f$ and aims for $\mathbf{p}_n$ next. The absolute values in the cost function can be handled by introducing auxiliary variables and extra constraints, according to the following principle: using an auxiliary variable s, the problem $\min |z|$ is equivalent to

$$\begin{aligned} &\min s \\ &\text{s.t. } z \leq s \\ &\quad -z \leq s. \end{aligned}$$

In the SEC demo, we used $T = 6$, corresponding to an effective planning length of 1 minute. All weights in the cost functions (10) and (11) where set to 1. In case the double integrator model was used, we also added a small input regularization term $\sum_{i=0}^{T-1} r|\mathbf{u}_i|$ with $r = 10^{-4}$ to produce smooth trajectories.

Velocity Bounds— To ensure that the planned trajectory respects the velocity limits of the vehicle, we added the following maximum and minimum speed constraints:

$\forall i \in [0 \dots T-1]$, $\forall k \in [1 \dots K]$:

$$\begin{array}{rcl} \dot{x}_i \sin\left(\frac{2\pi k}{K}\right) + \dot{y}_i \cos\left(\frac{2\pi k}{K}\right) & \leq & v_{\max} \\ \dot{x}_i \sin\left(\frac{2\pi k}{K}\right) + \dot{y}_i \cos\left(\frac{2\pi k}{K}\right) & \geq & v_{\min} - M c_{ik} \\ \sum_{k=1}^{K} c_{ik} & \leq & K-1 \\ c_{ik} & \in & \{0,1\} \end{array} \tag{12}$$

For the reference velocity model, these constraints were formulated in terms of the input commands $\dot{x}_{cmd,i}$ and $\dot{y}_{cmd,i}$ instead. The maximum speed bound is thus approximated by constraining the inertial velocity vector to lie inside a K-sided polygon. Using the binary variables c_{ik} and principle (6), the minimum speed requirement is captured by constraining the velocity vector to lie *outside* a (smaller) K-sided polygon.

In our software, we used $K = 32$. The minimum and maximum bounds were respectively set to $v_{\min} = 400$ fps and $v_{\max} = 450$ fps. To avoid infeasible problems in the event the actual ground speed fell outside these bounds (e.g. because of wind gusts), the latter were adapted accordingly.

Acceleration Bounds—As mentioned before, both models assume homogeneous dynamics in the x- and y-coordinates, thus ignoring differences in lateral and longitudinal dynamics. To correct for this, we added linear constraints that capture limits on turn rate and on forward and lateral acceleration. When flying at a more or less constant speed, the following acceleration constraints were sufficient [8]:

$\forall i \in [0 \dots T-1]$, $\forall k \in [1 \dots K]$:

$$\ddot{x}_i \sin\left(\tfrac{2\pi k}{K}\right) + \ddot{y}_i \cos\left(\tfrac{2\pi k}{K}\right) \leq a_{lat} \tag{13}$$

for the double integrator model, and

$\forall i \in [1 \dots T-1]$, $\forall k \in [1 \dots K]$:

$$\begin{array}{rcl} (\dot{x}_{cmd,i} - \dot{x}_{cmd,i-1}) \sin\left(\frac{2\pi k}{K}\right) & + & \\ (\dot{x}_{cmd,i} - \dot{x}_{cmd,i-1}) \cos\left(\frac{2\pi k}{K}\right) & \leq & a_{lat}\Delta t \end{array} \tag{14}$$

for the reference velocity model. We set the lateral acceleration bound $a_{lat} = 18.1$ ft^2/s, corresponding to a maximum turn rate of $\omega_{\max} = a_{lat}/v_{\min} = 2.6$ deg/s at $v_{\min} = 400$ fps.

When the velocity is allowed to change, however, these inequalities overestimate the available forward acceleration, which was limited to $a_{fwd} = 5.0$ ft^2/s. Therefore, to distinguish between forward and lateral acceleration, we added the following constraints:

$\forall i \in [0 \dots T-1]$, $\forall k \in [1 \dots K]$:

$$\begin{array}{rcl} (\ddot{x}_i + \alpha v_0^{-1} \dot{y}_i) \sin\left(\frac{2\pi k}{K}\right) & + & \\ (\ddot{y}_i - \alpha v_0^{-1} \dot{x}_i) \cos\left(\frac{2\pi k}{K}\right) & \leq & \beta a_{lat} \\ (\ddot{x}_i - \alpha v_0^{-1} \dot{y}_i) \sin\left(\frac{2\pi k}{K}\right) & + & \\ (\ddot{y}_i + \alpha v_0^{-1} \dot{x}_i) \cos\left(\frac{2\pi k}{K}\right) & \leq & \beta a_{lat} \end{array} \tag{15}$$

for the double integrator model, and

$\forall i \in [1 \dots T-1]$, $\forall k \in [1 \dots K]$:

$$\begin{array}{rcl} (\dot{x}_{cmd,i} - \dot{x}_{cmd,i-1} + \alpha v_0^{-1} \dot{y}_i \Delta t) \sin\left(\frac{2\pi k}{K}\right) & + & \\ (\dot{y}_{cmd,i} - \dot{y}_{cmd,i-1} - \alpha v_0^{-1} \dot{x}_i \Delta t) \cos\left(\frac{2\pi k}{K}\right) & \leq & \beta a_{lat}\Delta t \\ (\dot{x}_{cmd,i} - \dot{x}_{cmd,i-1} - \alpha v_0^{-1} \dot{y}_i \Delta t) \sin\left(\frac{2\pi k}{K}\right) & + & \\ (\dot{y}_{cmd,i} - \dot{y}_{cmd,i-1} + \alpha v_0^{-1} \dot{x}_i \Delta t) \cos\left(\frac{2\pi k}{K}\right) & \leq & \beta a_{lat}\Delta t \end{array} \tag{16}$$

for the reference velocity model. Here, v_0 is the current absolute ground speed, $\alpha = (a_{lat}^2 - a_{fwd}^2)/(2a_{fwd}) = 30.4$ fps^2/s, and $\beta = \sqrt{\alpha^2 + a_{lat}^2} = 35.4$ fps^2/s.

These inequalities describe the intersection of two circles in which the inertial acceleration vector has to lie. The short axis of this intersection has length $2a_{fwd}$ and is aligned with the velocity vector at the first time step. The long axis captures the larger lateral acceleration bound and has length $2a_{lat}$. Hence, this intersection approximates the dynamically feasible acceleration profile at the initial time step. More details about the derivation of these constraints can be found in [12].

Obstacle Avoidance—In our demonstration, we only considered rectangular no-fly zones aligned with the North-East coordinate frame. As mentioned before, only the zones lying inside the current detection region $\mathcal{D}_0$ of the aircraft needed to be accounted for. Denoting these obstacles by $j = 1 \dots J$ and describing them by their lower left (i.e. southwest) corner $(x_{\min,j}, y_{\min,j})$ and upper right (i.e. northeast) corner $(x_{\max,j}, y_{\max,j})$, the avoidance constraints $(x_i, y_i) \notin \mathcal{O}_0$ were formulated as follows [10]:

$\forall i \in [1 \dots T]$, $\forall j \in [1 \dots J]$:

$$\begin{array}{rcrcl} x_i & \leq & x_{\min,j} & + & M b_{i1} \\ -x_i & \leq & -x_{\max,j} & + & M b_{i2} \\ y_i & \leq & y_{\min,j} & + & M b_{i3} \\ -y_i & \leq & -y_{\max,j} & + & M b_{i4} \\ \sum_{k=1}^{4} b_{ij} & \leq & 3 & & \\ b_{ij} & \in & \{0,1\}. & & \end{array} \tag{17}$$

Applying principle (6) again, the last constraint ensures that at least one of the coordinate inequalities is active, thereby guaranteeing that the trajectory point (x_i, y_i) lies outside the rectangles $j = 1 \dots J$. Because the resulting trajectory consists of discrete waypoints, to prevent the UAV from cutting corners we enlarged the actual obstacles with a safety boundary of $d_{safe} = \frac{v_{\max}\Delta t}{\sqrt{2}} \approx 3200$ ft.

Loiter Constraints—As outlined in Section 3, safety can be guaranteed by ensuring that either a left or right loiter circle lying inside the detection region does not intersect with any of the obstacles $j = 1 \dots J$. As detailed in Ref. [11], we can express sample points along both circles as affine functions of the last state $\mathbf{x}_T$ in the planning horizon, and then introduce avoidance constraints similar to (17). As such, $\mathbf{x}_T$ is constrained to be ingress state to a *safe* loiter pattern.

Using an index k to indicate the sample points along the circles and introducing a binary variable d to select either the

right or left circle, the safe loiter constraints at each receding horizon iteration can then explicitly be expressed as:

$\forall k \in [1 \dots N]$, $\forall j \in [1 \dots J]$:

$$\begin{cases} x_T - \alpha_c (\cos k\theta_s - 1) v_{yT} - \alpha_c (\sin k\theta_s) v_{xT} \\ \qquad \leq \quad x_{\min,j} + Mb_{kj1} + Md \\ -x_T + \alpha_c (\cos k\theta_s - 1) v_{yT} + \alpha_c (\sin k\theta_s) v_{xT} \\ \qquad \leq -x_{\max,j} + Mb_{kj2} + Md \\ y_T - \alpha_c (\sin k\theta_s) v_{yT} + \alpha_c (\cos k\theta_s - 1) v_{xT} \\ \qquad \leq \quad y_{\min,j} + Mb_{kj3} + Md \\ -y_T + \alpha_c (\sin k\theta_s) v_{yT} - \alpha_c (\cos k\theta_s - 1) v_{xT} \\ \qquad \leq -y_{\max,j} + Mb_{kj4} + Md \end{cases}$$

AND

$$\begin{cases} x_T + \alpha_c (\cos k\theta_s - 1) v_{yT} + \alpha_c (\sin k\theta_s) v_{xT} \\ \qquad \leq \quad x_{\min,j} + Mb_{kj1} + M(1-d) \\ -x_T - \alpha_c (\cos k\theta_s - 1) v_{yT} - \alpha_c (\sin k\theta_s) v_{xT} \\ \qquad \leq -x_{\max,j} + Mb_{kj2} + M(1-d) \\ y_T + \alpha_c (\sin k\theta_s) v_{yT} - \alpha_c (\cos k\theta_s - 1) v_{xT} \\ \qquad \leq \quad y_{\min,j} + Mb_{kj3} + M(1-d) \\ -y_T - \alpha_c (\sin k\theta_s) v_{yT} + \alpha_c (\cos k\theta_s - 1) v_{xT} \\ \qquad \leq -y_{\max,j} + Mb_{kj4} + M(1-d) \end{cases}$$

$$\begin{cases} \sum_{l=1}^{4} b_{kjl} \quad \leq \quad 3 \\ b_{kjl},\ d \quad \in \quad \{0,1\} \end{cases}$$

Here $\theta_s = \frac{2\pi}{N}$ is the discretization angle around the circle. Again, because of the sampling procedure, the obstacle coordinates $(x_{\min,j}, y_{\min,j}, x_{\max,j}, y_{\max,j})$ are those obtained after enlarging the obstacles on all sides by a thickness d_{loiter}. Given a maximum turn radius of 1.9 mi, we used $N = 8$ and set $d_{loiter} = 0.6$ mi.

5. SOFTWARE IMPLEMENTATION

Our guidance software module is implemented in C++ and ILOG's Concert Technologies. To interface with the UAV avionics and guarantee hard real-time execution, it is integrated with Boeing's OCP software. The software runs on a Pentium 4 Linux laptop with 2.4 Ghz clock speed that is mounted in the aircraft, and interacts with the UAV avionics through a set of pre-defined command variables. Through the OCP interface the laptop receives GPS, ground speed and turn rate data, among other, at a rate of 20 Hz. The guidance module itself, however, only runs at 1 Hz. It consists of three submodules: a pre-processing step, an optimization step, and a post-processing step, which we will now discuss in more detail.

Pre-Processing

The pre-processing routine is called every second and determines all parameters of the MILP problem. It subsequently:

- Selects the correct LTI model,
- Estimates the initial state for the current planning horizon,
- Determines the relevant obstacles,
- Enlarges the obstacles with the appropriate safety band,
- Determines the intermediate waypoint, and
- Selects the appropriate cost function.

In addition, for numerical stability purposes and to speed up the MILP optimization, all latitude/longitude position data and obstacle coordinates are transformed to an East-North axis frame in kilometer units with the current position of the aircraft as the origin. Accordingly, the ground velocity of the aircraft is scaled to km/s.

Optimization

The optimization step is implemented using the mathematical programming package CPLEX from ILOG. CPLEX contains state-of-the art routines for solving large MILPs and comes with Concert Technologies, a C++ based modeling language. Using the latter, a MILP problem can be encoded in a compact form that is similar to the mathematical representation of it.

An important feature of CPLEX is its optional limit on computation time, which is critical for a hard real-time system. We set this limit to 0.85 s and the optimality gap to 10^{-4}. After the allocated time has passed, CPLEX either returns a feasible solution within the optimality gap, a feasible solution outside the optimality gap, or no solution at all. The last situation occurs when either the MILP itself is infeasible, or when no feasible solution can be found in time (e.g. because the problem is too complex).

Ideally, by definition of the safety constraints, the trajectory planning problem remains feasible at all times. However, because of disturbances such as wind gusts, the initial velocity might fall outside the constraint bounds, or the vehicle might be blown off course to a position from where an obstacle-free MILP solution no longer exists. The first situation is easy to spot and can be resolved ahead of time by resetting the velocity bounds in the pre-processing step. Infeasibilities caused by obstacles, however, are harder to predict and resolve. In that case, the UAV should resort to its backup plan, consisting of the previous plan minus the first time step and the previous loiter circle.

If the control authority used in the MILP problem (i.e. the admissible acceleration and turn rate limits) is somewhat conservative w.r.t. the actual performance of the vehicle, robust trajectories can be designed. Then, in case the UAV gets blown off course to a state from which no feasible solution to the MILP exists, the aircraft can use its additional control authority to get back to feasibility within a few time steps. In our code we therefore use maximum acceleration and turn rate bounds that are smaller than the actual ones available to the waypoint controller. As such, we never experienced infeasibilities of more than 2 or 3 time steps.

Although the pre-processing step is repeated every second,

the optimization function is nominally only executed every 10 s. The reason for this is the ∼10 s time constant of the T-33, which makes a higher planning rate unnecessary. Only when a large disturbance or an additional obstacle is detected, or when the vehicle is in backup plan mode (i.e. when the last MILP problem was infeasible), the optimization routine is executed at the next second. This way, the available time slots can be occupied by computations required by the task scheduling and natural language interface routines.

Post-Processing

The post-processing routine performs the feasibility check by interpreting a CPLEX flag, and updates the current trajectory (i.e. the current waypoint list), the loiter direction and a backup plan counter accordingly. In the nominal case where a feasible solution is found, the variables of interest are the 6 new states of the planning horizon (i.e. the waypoint coordinates with corresponding velocity vectors) and the new loiter direction. The coordinates are first transformed back to the original Greenwich-referenced longitude and latitude axis frame and the velocity is rescaled to fps. Next, the old plan is flushed, replaced by the new one, and the counter is set to 1, pointing to the first entry in the waypoint/state list. That entry is then given to a waypoint controller that issues forward velocity and turn rate commands to the vehicle.

If no feasible solution is found, however, the remainder of the previous trajectory is used as a backup plan. In this case, the backup plan counter is increased by one, thus pointing to the next waypoint of the existing plan. If the counter exceeds 6, depending on the value of the loiter direction binary, the left or right loiter circle is initiated by issuing a "Set and Hold Turn Rate" command of 3 deg/s to the UAV. Notice that this is slightly more aggressive than the maximum 2.6 deg/s turn rate accounted for in the plan. It results in a smaller loiter circle, which introduces some robustness to perturbations along the trajectory. As mentioned before, as long as the vehicle remains in the backup plan mode, the MILP optimization is executed every second (but the counter only updated every 10s), until a new feasible plan is found.

6. RESULTS

Demonstration Scenario

Using the narrative outlined in Section 2, we designed a variety of sample scenarios that are depicted in Figure 4. The flight area is approximately 40 mi across (east to west along the northern boundary) and 30 mi wide (north to south along the western boundary). There are two pre-determined no-fly zones (listed as "NFZ1" and "NFZ2") and three potential pop-up threats (denoted by "P Obs 1," "P Obs 2," and "P Obs 3"), which can be activated during the demonstration. In addition, there are two mission task areas (labeled "Search Area Alpha" and "Search Area Bravo"). Each task area has three potential ingress conditions which can be selected by the FW

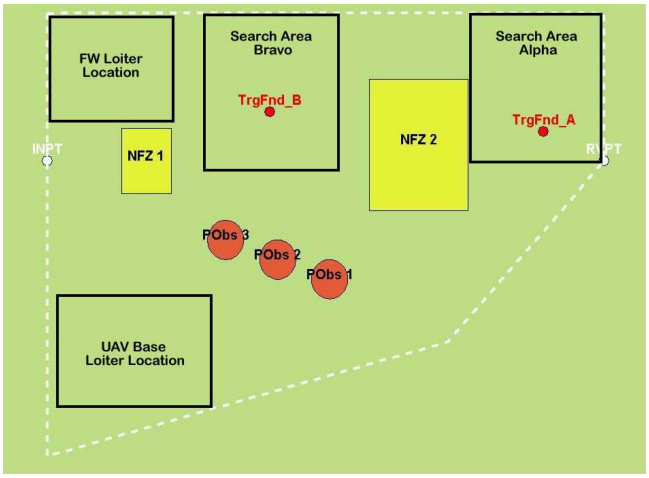

Figure 4. Sample Scenario Map for the MIT SEC Capstone Demonstration

WSO before the vehicle reaches the task area location. Each task area also includes a threat/target (denoted by "TrgFnd A" and "TrgFnd B") which the UAV searches for and locates during the mission. Finally, the UAV starts the mission from the UAV Base Loiter location in the southwest corner of the flight area, and the FW vehicle maintains a loiter pattern near the northern border until the target has been detected.

Simulation Results

To aid in the development of our guidance system, we built a real-time Simulation-In-the-Loop (SIL) test platform at MIT, which is shown in Figure 5. Besides the OCP, it included Boeing's DemoSim vehicle simulations for the UAV and FW aircraft, which ran on separate computers with a Link 16 communication interface.

Figure 5. MIT SEC Demonstration Simulation-In-the-Loop (SIL) laboratory setup

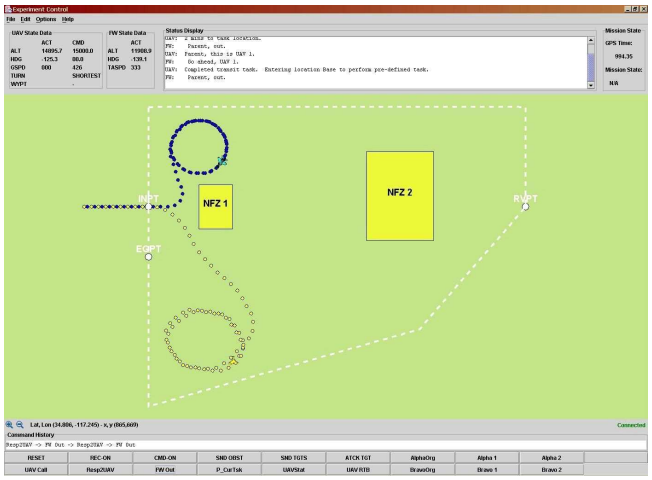

Figure 6. SIL Test 1 - Initialization of the SEC demonstration: the UAV (in yellow) enters a loiter pattern.

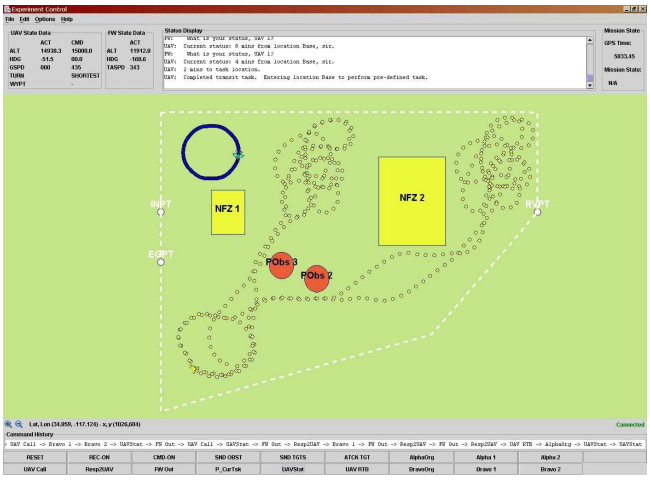

Figure 8. SIL Test 3 - Simulated flight with two consecutive search missions.

Figure 6 shows one of the many initialization tests. As the FW aircraft and UAV approach the flight area, the demonstration software is initialized: the UAV automatically flies to the UAV Base Loiter Location, where it remains until it is commanded another task. The loiter task itself is defined as a series of six waypoints with a fixed ingress location and heading. As can be seen from the picture, the UAV successfully avoids NFZ1 while flying to the loiter area.

Next, Figure 7 shows one of the pop-up obstacle avoidance tests used to verify the safety guarantees of the MILP trajectory planning algorithm. In this test, two pop-up obstacles were placed into the demonstration area as the UAV was en route to Search Area Alpha. The resulting trajectory highlights the MILP algorithm's ability to develop safe, dynamically feasible paths for the vehicle after unexpected changes to the environment right outside its detection radius. For example, when the UAV is south of the first pop-up obstacle (designated "P Obs 3"), the second one (designated "P Obs 1") is inserted, causing the UAV to immediately turn left and proceed northeast over it. After passing the pop-up obstacles, the vehicle levels out and flies at a safe distance from No-Fly Zone 2 (NFZ 2) before turning north to enter Task Area Alpha to perform a search task.

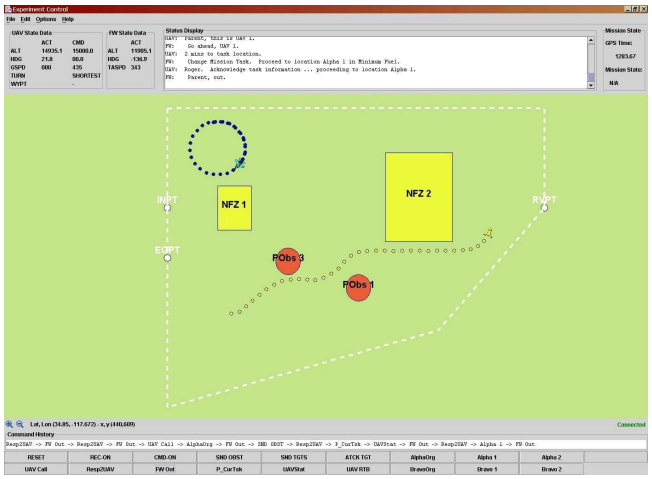

Figure 7. SIL Test 2 - Pop-up obstacle test: the UAV safely avoids both pop-up threats.

Figure 8 depicts a test where the UAV was commanded to fly two consecutive missions from the UAV Base Loiter Location. It shows the UAV's coverage over both search areas during the search task portions of the mission. Notice that the two search patterns are almost identical: the same task defining waypoint sequence (relative to the ingress position of the task area) was used in both search tasks, showing that the MILP guidance approach can accurately track waypoint plans. In addition, the UAV safely avoids two pop-up obstacles en route to each search area location.

Flight Test Setup

During the SEC Capstone Demonstration in late June 2004, multiple F-15/T-33 sorties were flown at NASA Dryden. Figure 9 shows a system level diagram of the flight experiment setup. During the test, the main role of the T-33's two person crew was to fly the vehicle to the demonstration area, activate our software and manage the vehicle in the event of failures. In addition, for technical reasons, the T-33 pilot executed the forward velocity commands produced by the MILP guidance algorithm. The turn rate, however, was directly commanded by the laptop.

Although the FW WSO was only able to select UAV-tasks from a pre-defined list of options, the actual mission was not pre-planned. The T-33's rear-seat operator (nicknamed

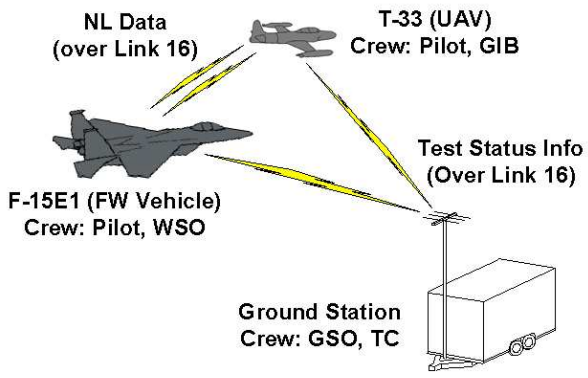

Figure 9. Flight experiment system level diagram

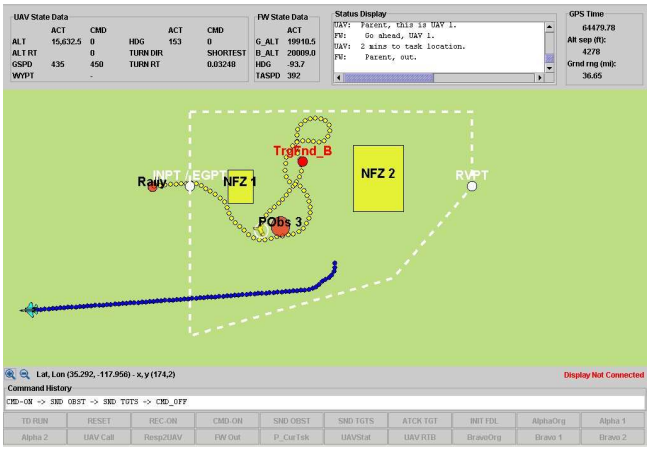

Figure 10. SEC Flight Test: UAV/T-33 (in yellow) with simulated F-15

the "Guy-In-Back" or "GIB") observed the progress of the demonstration and a Ground Station Operator (GSO) added pop-up obstacles via experiment key commands, introducing another degree of uncertainty into the mission. Namely, although the possible locations of the mission task areas and pop-up obstacles were pre-defined, they were selected randomly in real-time during each flight experiment. The experiment Test Coordinator (TC) monitored the demonstration from the ground station and communicated status information about the local airspace to the pilots.

Flight Test with Simulated F-15

In the first test, shown in Figure 10, the T-33/UAV flew a mission with a simulated F-15 while the Ground Station Operator issued the natural language commands. The flight took place in the morning of Thursday, June 17th, 2004, with a wind of 5 to 10 knots blowing from the southwest corner of the flight area (at a heading of 220 to 240 degrees).

Flight Account— The UAV started west of the ingress point (labelled "INPT / EGPT" in Figure 10) with a heading of 090. After the GIB initialized the mission software, the UAV began turning south to avoid NFZ 1. As it turned south, the vehicle appeared to fluctuate between a heading of 125 and 130. After it passed the lower left hand corner of NFZ 1, the Test Coordinator notified the T-33/UAV pilot and GIB that they had to change the flight card to avoid the southwestern corner of the flight area (because of a real-time flight area restriction). As such, when the vehicle was approximately two miles SSW of NFZ 1, the GSO commanded the UAV to proceed to Location Bravo 1. The UAV responded and began turning left toward Task Area Bravo. Within two minutes of this command, the GSO inserted Pop-up Obstacle 3 into the test area. At this point, the vehicle had a heading between 070 and 080 and was approximately four miles from the obstacle. It began to turn right to avoid the obstacle and flew along its southern boundary, successfully avoiding it. After passing the obstacle, the UAV proceeded to turn north toward the Bravo 1 location and initiated the search pattern. As the vehicle was facing SSE, it was near the target location and the GSO inserted the target into the environment. The UAV sent the proper status messages to the operator and the GSO commanded the vehicle to return to base. It began to turn right and safely flew southwest around Pop-up Obstacle 3. Since the southwest airspace was off-limits, however, the UAV was not permitted to return to its Base Loiter location and the mission was ended.

Flight Evaluation— During this test flight, the vehicle was commanded to fly to the Bravo Task Area before reaching the UAV Base Loiter location. This verified the flexibility of our mission software and the ability of an operator to easily change the mission goals and needs in real-time. In addition, the pop-up obstacle was inserted when it was already located within the vehicle's detection radius. Regardless, the MILP trajectory planning software calculated a solution for the vehicle, allowing it to safely avoid the obstacle. The flight test marked the first time a MILP-based onboard guidance system was used to control a UAV.

Flight Test with actual F-15

In a second test, shown in Figure 11, the T-33/UAV flew a successful mission with the F-15 WSO issuing natural language commands. This flight took place in the afternoon on Wednesday, June 23rd, 2004, with a SW wind (220-240 deg) between 10 and 15 knots.

Flight Account— The UAV again started west of the ingress point with a heading of 090, and began turning south to avoid NFZ 1 after the GIB started the experiment software. As it turned south, the vehicle steadily moved to the UAV Base Location. After reaching the loiter location, it started to turn right to a heading of 270 at which point the F-15 WSO commanded it to fly to Task Area Bravo. The UAV responded and began turning northward, thereby avoiding Pop-up Obstacle 3 and notifying the F-15 WSO that it detected a threat. As it approached the task area, the UAV notified the F-15

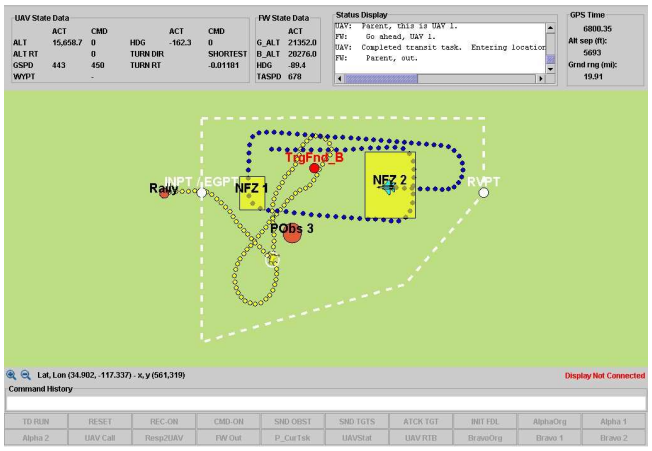

Figure 11. SEC Flight Test - UAV/T-33 (in yellow) and actual F-15 providing natural language commands.

WSO that it was two minutes from the ingress point it was initially given. At that time, the WSO commanded the UAV to change the entrance location. It responded and proceeded to the new ingress point.

After reaching the task area, the UAV notified the F-15 WSO and began its search pattern. As it turned left toward the southern boundary of the search area, the GSO inserted Target B into the environment. The UAV notified the F-15 WSO and sent an image message to the F-15. The F-15 WSO then commanded the UAV to return to base. The UAV acknowledged the command and flew back to its loiter location, thereby avoiding obstacles and providing status notifications to the F-15 WSO along the way. The demonstration was ended when the UAV successfully returned to the UAV loiter area.

Flight Evaluation— During this flight, the MILP trajectory planning software again successfully managed to account for no-fly zones and for changing mission objectives. The pop-up obstacle was inserted well beyond the detection radius, such that the UAV had adequate time to plan around it. The flight test marked the first time that a MILP-based guidance system was used to control a UAV in coordination with a manned vehicle. In addition, it also marked the first time that a natural language interface was used by a manned vehicle to command a UAV in real-time.

7. Conclusion

This paper discussed the implementation of a UAV guidance system based on Mixed Integer Linear Programming that was developed for the Capstone Demonstration of DARPA's Software Enabled Control Program. The guidance software was part of a mission system allowing a fixed wing operator to give high level tasks to a UAV in natural language. Given task waypoints, MILP was used to compute safe, dynamically feasible trajectories in real-time through a partially-known environment. The detailed MILP formulation using safety loiter circles was presented along with practical decisions about the implementation in C++/CPLEX and the integration with Boeing's OCP platform.

Simulation and June 2004 flight test results were presented, which mark the first time an on-board MILP-based guidance system was used to control a UAV. These results highlight the performance, flexibility and robustness of the MILP approach to online trajectory planning in cluttered environments and the feasibility of its implementation on future UAV systems. Currently we are extending this work to platforms with multiple unmanned vehicles for which real-time decentralized trajectory planning strategies are developed that exhibit similar safety guarantees.

Acknowledgements

The authors would like to thank Timothy Espey, Brian Mendel, Dr. Jim Paunicka, and Jared Rosson from Boeing Phantom Works, and Yoshiaki Kuwata from MIT for their help with this project. The research was funded by DARPA SEC contract F33615-01-C-1850.

References

[1] *ILOG CPLEX 9.0 User's guide*. ILOG, 2003.

[2] J. Bellingham, A. Richards, and J. How. Receding horizon control of autonomous aerial vehicles. In *Proc. 2002 American Control Conference*, Anchorage, AK, May 2002.

[3] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999. Pergamon/Elsevier Science, New York NY.

[4] E. Craparo and E. Feron. Natural language processing in the control of unmanned aerial vehicles. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004.

[5] C. A. Floudas. *Nonlinear and Mixed-Integer Programming - Fundamentals and Applications*. Oxford University Press, 1995.

[6] B. Mettler, M. Valenti, T. Schouwenaars, Y. Kuwata, J. How, J. Paunicka, and E. Feron. Autonomous uav guidance build-up: Flight-test demonstration and evaluation plan. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Austin, TX, August 2003.

[7] J. Paunicka, B. Mendel, and D. Corman. The ocp - an open middleware solution for embedded systems. In *Proc. 2001 American Control Conference*, Arlington, VA, June 2001.

[8] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proc. 2002 American Control Conference*, Anchorage, AK, May 2002.

[9] E. Roche. Parsing with finite-state transducers. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press, 1997.

[10] T. Schouwenaars, B. DeMoor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *Proc. European Control Conference (ECC'01)*, Porto, Portugal, September 2001.

[11] T. Schouwenaars, J. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. In *Proc. 2004 American Control Conference*, Boston, MA, June 2004.

[12] T. Schouwenaars, B. Mettler, E. Feron, and J. How. Hybrid architecture for full-envelope autonomous rotorcraft guidance. In *59th Annual Forum of the American Helicopter Society*, Phoenix, AZ, May 2003.

[13] H. A. Taha. *Operations Research, An Introduction*. Macmillan Publishing Company, New York, 4th edition, 1987.

[14] M. Valenti, T. Schouwenaars, Y. Kuwata, E. Feron, J. How, and J. Paunicka. Implementation of a manned vehicle-uav mission system. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004.

[15] J. Wohletz, D. Castanon, and M. Curry. Closed-loop control for joint air operations. In *Proc. 2001 American Control Conference*, Arlington, VA, June 2001.

BIOGRAPHIES

Tom Schouwenaars *is a Research Assistant in the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. He obtained a M.Sc. degree in electrical engineering from the Catholic University of Leuven, Belgium, in 2001, and is currently a Ph.D. candidate in Aero- and Astronautics at MIT. He is a Francqui Fellow of the Belgian American Educational Foundation and was awarded the 2005 AIAA Unmanned Aerial Vehicles Graduate Award.*

Mario Valenti *is a Research Assistant in the Laboratory for Information and Decision Systems at MIT. He has B.Sc. degrees in Mechanical and Electrical Engineering from Temple University. For two years, he worked as a Flight Controls Engineer in the Integrated Defense Systems Division of the Boeing Company, where he is currently on a leave of absence. In June 2003, he received a M.Sc. in Electrical Engineering and is currently pursuing his Ph.D. in Electrical Engineering at MIT.*

Eric Feron *is an Associate Professor of Aeronautics and Astronautics at MIT, where he is affiliated with the Laboratory for Information and Decision Systems. He is a graduate from Ecole Polytechnique in Paris, and received his Ph.D. in Aero- and Astronautics from Stanford University in 1994.*

Jonathan How *is an Associate Professor of Aeronautics and Astronautics at MIT, where he is affiliated with the Aerospace Controls Laboratory. He graduated from the University of Toronto with a B.Sc. in Aerospace Engineering, and received the M.Sc. and Ph.D. degrees in Aero- and Astronautics from MIT in 1989 and 1993 respectively.*