JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 29, No. 2, March–April 2006

Linear Programming and Language Processing for Human/Unmanned-Aerial-Vehicle Team Missions

T. Schouwenaars,* M. Valenti,† E. Feron,‡ and J. How§
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
and
E. Roche¶
Teragram Corporation, Cambridge, Massachusetts 02138

This paper presents a manned-vehicle/unmanned-aerial-vehicle (UAV) mission system that enables an operator in a manned aircraft to issue mission level commands to an autonomous aircraft in real time. A natural language interface allows the manned and unmanned vehicle to communicate in languages understood by both agents. A task scheduler transforms the commands into a dynamic mission plan consisting of task waypoints. These are then given to a mixed-integer linear programming (MILP)-based trajectory optimizer, which safely guides the vehicle through a partially known environment in real time. The MILP trajectory planning formulation and its implementation are discussed in detail. Integrated simulation and June 2004 flight-test results that used an F-15 and an autonomous T-33 equipped with Boeing's unmanned combat air vehicle (UCAV) avionics package are presented. These activities were part of the Capstone Demonstration of the Defense Advanced Research Projects Agency-sponsored Software Enabled Control effort. The flight tests mark the first time that an onboard MILP-based guidance system was used to control a UAV. They also mark the first time that a natural language interface was used by a manned vehicle to task a UAV in real time.

I. Introduction

UNMANNED aerial vehicles (UAVs) are used by both military and civilian organizations in a number of applications.[1] Recent advances in guidance technologies have enabled some UAVs to execute simple mission tasks without human interaction. Many of these tasks are preplanned using reconnaissance or environment information. For example, air operations are executed according to an air tasking order, which can take up to 72 h to plan, task, and execute.[2] In volatile situations, however, information about the vehicle's operating environment can 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: the vehicle's guidance and mission planning systems must possess enough intelligence (and processing power) to recognize and react to changes in the operating conditions.

The complexity of the preceding problem increases when more than one agent is introduced. For example, if other autonomous agents are added to the mission scenario, then all vehicles must resolve information regarding the impending actions of the other vehicles. Similarly, if a manned agent is introduced, the autonomous vehicles must also possess the capability to effectively communicate and coordinate their actions with the manned vehicle. Most unmanned vehicles, however, do not exhibit this level of performance. Intelligent mission and guidance systems providing the flexibility and cooperative behavior needed to complete an entire mission autonomously are therefore a topic of active research.[3–6]

This paper discusses the development, implementation, and evaluation of such a system containing a manned vehicle and a UAV operating in a partially known environment. It enables the operators of the manned aircraft to issue tasks and mission-level commands to the unmanned aircraft in real time using a natural language interface. The latter translates English sentence commands from the crew to a set of codes understood by the UAV and vice versa. A task scheduler then transforms these commands into input data of an online trajectory optimization problem, which is formulated as a mixed-integer linear program. Mixed-integer linear programming (MILP) is a powerful optimization framework that extends continuous linear programming to include binary or integer decision variables.[7] These variables can be used to model logical constraints such as obstacle and collision-avoidance rules, whereas the dynamic and kinematic properties of the vehicle are formulated as continuous constraints.

Under the Defense Advanced Research Projects Agency (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.[8] 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 that 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 online, that is, as the mission unfolds. Thanks to the increase in computer speed and implementation of powerful state-of-the-art algorithms in software packages such as CPLEX,[9] MILP has become a feasible option for real-time path planning, as demonstrated by the results discussed in this paper.

The overall mission system thus transforms the natural language commands of the manned aircraft operators into a mathematical

Presented as Paper 2004-5142 at the AIAA Guidance Navigation and Control Conference, Providence, RI, 16–19 August 2004; received 8 September 2004; revision received 15 July 2005; accepted for publication 19 July 2005.

*Ph.D. Candidate, Department of Aeronautics and Astronautics, Laboratory for Information and Decision Systems; toms@mit.edu. Student Member AIAA.

†Ph.D. Candidate, Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems; valenti@mit.edu. Student Member AIAA.

‡Associate Professor of Aeronautics and Astronautics, Laboratory for Information and Decision Systems; feron@mit.edu. Associate Fellow AIAA.

§Associate Professor of Aeronautics and Astronautics, Aerospace Controls Laboratory; jhow@mit.edu. Associate Fellow AIAA.

¶Principal, Executive Director; roche@teragram.com.

programming problem producing real-time trajectories that implement the dynamic mission plan of the UAV. A number of challenges had to be overcome during the development of this system. First, the mechanism allowing both vehicles to communicate with one another needed to be designed. Second, the UAV guidance technology had to be made robust to changes and threats in the vehicle's environment—including changing wind conditions, no-fly zones, and other obstacles—and produce safe trajectories through the partially known environment. Third, because the system was intended for real-time missions, all developed algorithms needed to reach a solution and resolve any unexpected issues reliably in a predefined period of time.

As part of the SEC Program, the system was implemented on a testbed consisting of an F-15, acting as the manned aircraft, and a T-33 augmented with Boeing's UCAV avionics package, acting as the autonomous vehicle. During the SEC Capstone Demonstration in June 2004, it was successfully flight tested at the NASA Dryden Flight Research Center. These flight tests mark the first time that a natural language interface was used by a manned vehicle to task a UAV in real time and the first time that a MILP-based guidance system was used to control a UAV.

The paper is organized as follows. Section II gives an overview of the SEC experiment and the associated technology development. Section III describes the natural language interface, and Sec. IV discusses the task scheduling and communication components. The basic safe trajectory planning problem is outlined in Sec. V. Section VI then presents the corresponding MILP formulation in detail, and Sec. VII discusses the real-time software implementation and associated practical engineering decisions. Simulation and flight-test results are then presented in Sec. VIII.

II. Experiment and Technology Overview

A. Mission Scenario

As originally discussed in Ref. 10, as part of the DARPA-sponsored Software Enabled Control program, our team was tasked with developing a mission system and flight-test scenario that exhibited UAV technology developed at Massachusetts Institute of Technology (MIT). For this demonstration, two flight assets were available: a Boeing F-15E fighter jet and a Lockheed T-33 trainer fighter jet 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 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)[11] and loaded onto a laptop fitted in each aircraft. The OCP software provided an aircraft interface that included the following abilities: 1) send and receive state and user-defined data between both aircraft using a Link-16 communications interface; 2) receive the current vehicle state data; 3) send a set of predefined 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 4) memory storage and time frame execution.

Given these demonstration resources, a mission scenario (shown in Fig. 1) was developed in which the UAV performs tasks in support of the FW vehicle.

1. 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. Because the environment is only partially known, there can be threats to both the manned and unmanned aircraft.

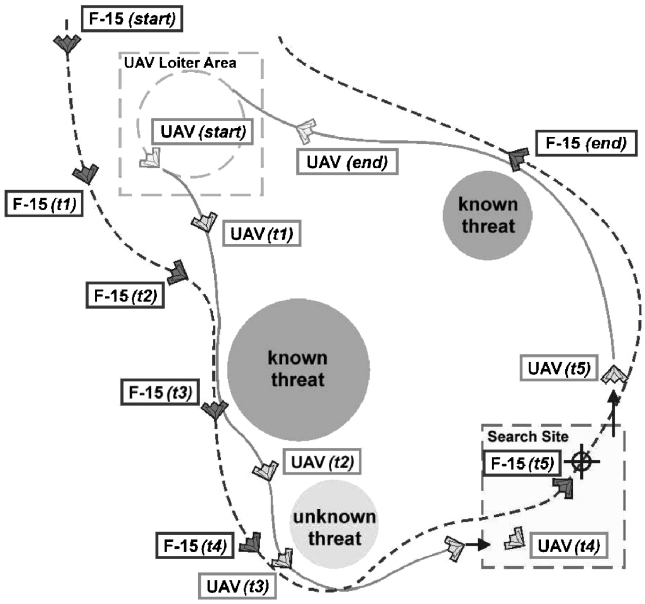

Fig. 1 Overview of the MIT flight experiment.

2. Starting Condition

The UAV will start in a predefined 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 can arise en route to the search site, which is currently unknown.

3. 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 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, it notifies the FW vehicle of its location. At this point, the FW 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 vehicle of its location, and begins its search for the target.

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

4. Exit Conditions

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

B. Technology Development

The preceding 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 from the humans onboard the FW vehicle into data the UAV

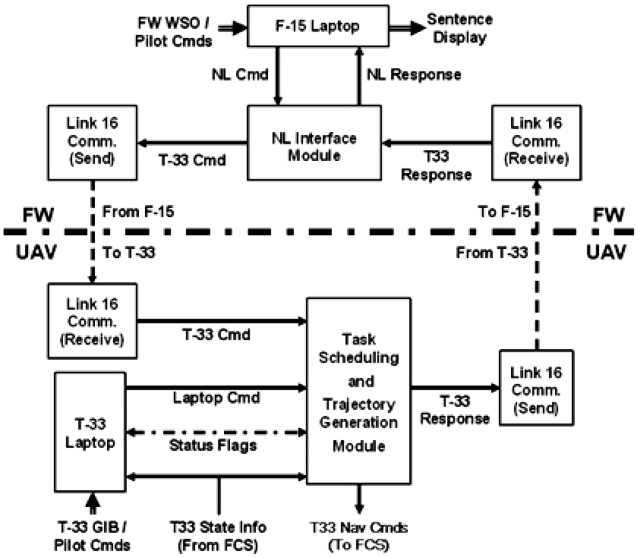

Fig. 2 Block diagram of the MIT SEC Capstone Demonstration system: FCS, flight control system; GIB, guy-in-back (i.e., the rear-seat operator).

can understand and use and vice versa. It enables the FW WSO to give high-level mission commands to the UAV in English, for example, "search this region for threats," rather than low-level guidance commands such as "turn left" or "speed up." As such, the natural language interface is aimed at minimizing the workload of the FW WSO when interacting with the computer-based UAV.

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 mission tasks that were developed included flying to a waypoint X, entering a loiter pattern, and performing a search pattern. 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 and obstacle avoidance information, and acknowledgement messages.

3. MILP-based Trajectory Generation

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 MILP. Because in the mission scenario the environment is only partially known and is explored in real time, the MILP guidance algorithm uses a receding-horizon planning strategy, allowing for online trajectory computation.

Each of these components addresses a capability required to perform the preceding mission. Figure 2 shows a block diagram representation of the integrated FW and UAV demonstration system. In the following sections, the development and integration of the three technologies is discussed. The focus, however, is placed on the trajectory generation module.

III. Natural Language Parsing and Interfacing

The main goal of using a natural language interface (NLI) for interacting with a computer-based system is to minimize the workload on the operator. Using normal English sentence commands indeed allows the FW WSO or pilot to communicate efficiently and effectively with the UAV, as if it were a human wingman. The NLI module developed for the demonstration consists of two major components. The first one takes sentence commands from the FW WSO and turns them into a coded command that is sent to the UAV over Link-16. The second component takes a coded command set from the UAV and converts it into a natural language response for the FW WSO to interpret.

A sample dialogue between the FW WSO and the UAV could be as follows, in which the UAV is commanded to search a predefined region containing a potential threat:

FW: "UAV 7, this is Eagle 3."

UAV: "Go ahead, Eagle 3."

FW: "Add new mission task. Proceed to location Echo-Charlie 5 in minimum time. Search this region for threats and wait for further instructions after the task is completed."

UAV: "Roger. Acknowledge task information—proceeding to location Echo-Charlie 5."

FW: "Eagle 3, out."

The NLI module analyzes the natural sentences produced by the FW WSO using parsing, which is the process of converting an input sentence, for example, "Proceed to location Echo-Charlie 5 in minimum time," into a formal representation. The latter is typically a tree structure, which can in turn be translated into an explicit formal command. In our system, parsing consists of first applying entity extraction to all of the individual concepts (e.g., "Eagle 3" or "Echo-Charlie 5") and then combining these concepts through cascades of finite-state transducers using techniques derived from those described in Ref. 12.

As the vocabulary of the SEC experiment is much smaller than for generic applications, the level of ambiguity is reduced, which makes parsing easier than on more open text. However, compared to other information processing tasks, this deployment required a particular emphasis on the safety of the parsing process. The stability of the run-time module was achieved by shifting the complexity of the system toward the off-line model compilation phase (for which there are much less stringent stability requirements). This makes the run-time process much simpler. In addition, the run-time process consists mostly of finite-state operations whose algorithm can be proven correct and for which the input finite-state machines can be checked for particular formal properties. This approach has the additional benefit of providing a very efficient processing time, which can be bounded explicitly.

Because of budgetary constraints, our team was unable to incorporate voice recognition in the system. Instead, a set of useful commands available to the FW WSO through predefined experiment keys on the F-15 laptop was chosen. When an experiment key is pressed, the associated sentence is sent to the NLI module. It is then parsed and converted into a nine-number code used by the UAV as an input command. This code uses the following protocol:

Message Description
Cmd ID: <long integer>
Cmd Data Words 1-8: <double>

The command identification (Cmd ID) value denotes the type of command sent between the vehicles, whereas the command data words contain the actual information. For example, Cmd ID 102 can represent the "command acknowledge" data set, and Cmd ID 106 can represent the "new/change current task" data set. The Cmd data words corresponding to Cmd ID 102 might then consist of a coded representation of "Proceeding to location Echo-Charlie 5." Because of user bandwidth limitations in the demonstration, each command word identifies a maximum of eight data words.

IV. Task Scheduling and Communications Interfacing

The task scheduling and communications processing components are designed to centralize all of the UAV's mission processing in one module. Together with the natural language interface, it provides flexibility for an operator to insert and change mission tasks during the operation. The UAV software keeps track of the mission tasks, waypoint locations, and known obstacles to pass on to the guidance algorithm.

The communications processing component provides the FW WSO with the authority to send task commands and receive status updates, threat or obstacle avoidance information, and acknowledgement messages. It also provides the ground operators

monitoring the UAV during the demonstration with the ability to override the guidance system in the event of an emergency or error. The system sends threat and override information to the FW WSO before any status or update information in an effort to send the most important data relevant to the demonstration before any auxiliary information. Input/output data are processed every 1-Hz frame before the task planner and guidance step to ensure that the most up-to-date information is used by the UAV trajectory planner.

The task scheduling component allows a user to plan a number of tasks using a predefined list or as programmed during a mission. Because many missions are preplanned, the system allows an operator to initiate a predefined mission task or to modify or create a mission plan by entering specific task parameters. The list of mission tasks includes the following: Fly to Waypoint X, Loiter Pattern, Search Pattern, Classify Target, Attack Target, Battle Damage Assessment, and Return to Base. For each of these task options, the user must provide the ingress and egress conditions and the size and location of a rectangular task area, given by the lower-left and upper-right coordinates. In addition, he or she has the option of providing (in real time via the NLI) the optimization metric used by the trajectory generation algorithm (i.e., minimum time, minimum fuel, or the amount of time to finish the task).

Next, the operator can either give the vehicle a new task or change the current task it is performing. A "New Task" command is added to the end of the UAV task list and is executed after all of the tasks currently in the scheduler have been completed. A "Change Task" command, on the other hand, modifies the current task performed by the UAV. Once a task is completed, it is removed from the list. After each of these actions, an acknowledgment is sent to the FW WSO, and the updated task information is included in the data sent to the trajectory generation module.

To reduce the complexity of the demonstration system, only the current task could be modified, although future versions of this system will have the capability to change any of the tasks in the scheduler. Furthermore, because of communication link bandwidth constraints, the FW WSO did not have the capability to define a new task or adjust parameters in the current task manually. Instead, he was able to command the vehicle to perform tasks from a predefined library using the experiment keys on the FW laptop.

V. Basic Trajectory Planning Problem

After the natural language interface and task scheduling components have converted the mission steps into a series of tasks for the UAV to perform, the trajectory generation module guides the vehicle from one task location to the next, that is, 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. For the demonstration, two-dimensional scenarios were considered in which no-fly zones or "obstacles" are detected while the mission is carried out, but such that the environment is always fully characterized inside a certain detection region $\mathcal{D}$ around the aircraft. The demonstration scenarios used a circular region of radius 9 miles and assumed that all obstacles $\mathcal{O}$ within that radius were static. The resulting formulation can, however, be easily generalized to account for any detection shape, such as a radar cone, and for unknown areas within that shape.

Because trajectories must be dynamically feasible, the UAV 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 $(\boldsymbol{A}, \boldsymbol{B})$ in an inertial two-dimensional coordinate frame (east-north). As such, the state vector $\boldsymbol{x}$ consists of the east-north position (x, y) and corresponding inertial velocity $(\dot{x}, \dot{y})$. Depending on the particular model, the input vector $\boldsymbol{u}$ is an inertial acceleration or reference velocity vector. In both cases, however, combined with additional linear inequalities in $\boldsymbol{x}$ and $\boldsymbol{u}$, the state-space model must capture the closed-loop dynamics that result from augmenting the vehicle with a waypoint tracking controller.

Because the environment is only partially known and further explored in real time, a receding-horizon planning strategy is used to guide the vehicle towards the desired destination. The latter is denoted by $\boldsymbol{x}_f$ and is an ingress or 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 $\boldsymbol{x}_0 = (x_0, y_0, \dot{x}_0, \dot{y}_0)$ in the optimization problem should be an estimate $\boldsymbol{x}_{\text{estim}}$ 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:

$$\min_{x_i, u_i} J_T = \sum_{i=0}^{T-1} f_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_f) + f_T(\boldsymbol{x}_T, \boldsymbol{x}_f) \tag{1}$$

subject to

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

The objective function (1) consists of stage costs $f_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_f)$ corresponding to each time step i and a terminal cost term $f(\boldsymbol{x}_T, \boldsymbol{x}_f)$ that accounts for an estimate of the cost-to-go from the last state $\boldsymbol{x}_T$ in the planning horizon to the goal state $\boldsymbol{x}_f$. The sets $\mathcal{X}_0$ and $\mathcal{U}_0$ represent the (possibly nonconvex) constraints on the vehicle dynamics and kinematics, such as bounds on velocity, acceleration, and turn rate. 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 $i = 0$. Note that they are assumed to hold for $\boldsymbol{x}_0$; if not, the trajectory optimization problem would be infeasible from the start.

As demonstrated in Ref. 8, however, despite the detection region and avoidance constraints, the preceding receding-horizon strategy has no safety guarantees regarding avoidance of obstacles in the future. Namely, the algorithm might fail to provide a solution in future time steps because of 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 future receding horizon iteration.

In Ref. 8, a safe receding-horizon scheme was therefore proposed based on maintaining a known feasible trajectory from the final state $\boldsymbol{x}_T$ in the current planning horizon towards an obstacle-free holding pattern. The latter must lie in the region $\mathcal{D}_0$ 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 holding pattern can then always serve as an a priori safe backup or "rescue" plan. For the SEC demonstration in particular, the final state $\boldsymbol{x}_T$ was constrained to be an ingress state to a right or left turning loiter circle.

VI. Mixed-Integer Linear Programming Formulation

The optimal safe trajectory planning problem just outlined lends itself well to be formulated as a mixed-integer linear program. MILP

is a powerful mathematical programming framework that allows inclusion of integer variables and discrete logic in a continuous linear optimization problem.[7] It is commonly used in Operations Research[13] and has more recently been introduced to the field of hybrid systems[14] and trajectory optimization.[15] 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. The following applies the MILP framework to the trajectory optimization problem (1) and (2).

A. 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. A piecewise linear first-order approximation was deemed to be sufficient for guidance purposes. The time constant and dc gain of the transfer function were identified for a discrete set of forward velocities (from 350 to 500 fps with a resolution of 10 fps) and stored 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, linear time-invariant (LTI) modes scheduled around the initial velocity.

Taking the desired inertial velocity as input then gives the continuous-time state-space model:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1/\tau_l & 0 \\ 0 & 0 & 0 & -1/\tau_l \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ k_l/\tau_l & 0 \\ 0 & k_l/\tau_l \end{bmatrix} \begin{bmatrix} \dot{x}_{\text{cmd}}(t) \\ \dot{y}_{\text{cmd}}(t) \end{bmatrix} \tag{3}$$

where τ_l is the time constant and k_l is the gain corresponding to the lth 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. To correct for this, subsection VI.C introduces additional constraints on the state and input vectors.

For optimization purposes, the model in Eq. (3) was discretized using the bilinear transform to give

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \dot{x}_{i+1} \\ \dot{y}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{\Delta t}{2}\left(1+\frac{2\tau_l-\Delta t}{2\tau_l+\Delta t}\right) & 0 \\ 0 & 1 & 0 & \frac{\Delta t}{2}\left(1+\frac{2\tau_l-\Delta t}{2\tau_l+\Delta t}\right) \\ 0 & 0 & \frac{2\tau_l-\Delta t}{2\tau_l+\Delta t} & 0 \\ 0 & 0 & 0 & \frac{2\tau_l-\Delta t}{2\tau_l+\Delta t} \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} + \begin{bmatrix} \frac{k_l(\Delta t)^2}{2\tau_l+\Delta t} & 0 \\ 0 & \frac{k_l(\Delta t)^2}{2\tau_l+\Delta t} \\ \frac{2k_l\Delta t}{2\tau_l+\Delta t} & 0 \\ 0 & \frac{2k_l\Delta t}{2\tau_l+\Delta t} \end{bmatrix} \begin{bmatrix} \dot{x}_{\text{cmd},i} \\ \dot{y}_{\text{cmd},i} \end{bmatrix} \tag{4}$$

Because the typical time constant of the T-33 velocity response was around 9.7 s, a time step of $\Delta t = 10$ s was used. With the additional constraints from subsection VI.C, this model produced good results for tasks requiring intensive waypoint tracking and sharp turns (i.e., the loiter and search tasks). For less aggressive trajectories with more or less constant speed, such as when transitioning between two task areas, a simpler double integrator model was used that does not distinguish between the different LTI modes:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \dot{x}_{i+1} \\ \dot{y}_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_i \\ \ddot{y}_i \end{bmatrix} \tag{5}$$

The discretization step was again set to $\Delta t = 10$ s.

Because it takes a certain time to compute the trajectory, the initial state $\boldsymbol{x}_0$ should be an estimate of the vehicle's state when the plan is actually implemented. In our case, the computation delay was approximately 1 s. In addition, a 1.2-s actuator delay needed be accounted for. To obtain an estimate of the initial state at the next iteration, the dynamics were thus propagated forward following the previous plan for 2.2 s.

B. Cost Function

Given the preceding models, the goal is to guide the UAV between waypoints in the fastest possible way, thereby 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. Because 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 Refs. 16 and 17 is not possible. Instead, the following heuristic was used.

In the case that 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 $\boldsymbol{p}_f = (x_f, y_f)$, the piecewise linear cost function

$$\min J = \sum_{i=0}^{T} -q\boldsymbol{v}_i'(\boldsymbol{p}_f - \boldsymbol{p}_{\text{estim}}) + r|\boldsymbol{p}_i - \boldsymbol{p}_f| \tag{6}$$

was used to design a fast trajectory between the initial position $\boldsymbol{p}_0 = \boldsymbol{p}_{\text{estim}} = (x_0, y_0)$ in the planning horizon and $\boldsymbol{p}_f$. The first term in this objective function tries to maximize the scalar product of the inertial velocity $\boldsymbol{v}_i = (\dot{x}_i, \dot{y}_i)$ 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 and turn it toward waypoint $\boldsymbol{p}_f$. In addition, the term $r|\boldsymbol{p}_i - \boldsymbol{p}_f|$ tries to minimize the one-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 $\boldsymbol{p}_f$ at a time step $T_f \leq T$ in the planning horizon, the cost function for the next iteration is split into

two parts:

$$\min \tilde{J} = \sum_{i=0}^{T_f-1} -q_f \boldsymbol{v}_i'(\boldsymbol{p}_f - \boldsymbol{p}_{\text{estim}}) + r_f|\boldsymbol{p}_i - \boldsymbol{p}_f| + \sum_{i=T_f}^{T} -q_n \boldsymbol{v}_i'(\boldsymbol{p}_n - \boldsymbol{p}_f) + r_n|\boldsymbol{p}_i - \boldsymbol{p}_n| \tag{7}$$

in which $\boldsymbol{p}_n$ is the next (intermediate) waypoint. The first $T_f - 1$ time steps are thus used to minimize the cost towards waypoint $\boldsymbol{p}_f$; the remaining steps aim at minimizing the cost towards the next waypoint $\boldsymbol{p}_n$. As a result, depending on the relative weighting, the MILP optimization will produce a trajectory that passes through or close by $\boldsymbol{p}_f$ and aims for $\boldsymbol{p}_n$ next. The SEC demonstration used $T = 6$, corresponding to an effective planning length of 1 min, and all weights in Eqs. (6) and (7) were set to 1. Note that the absolute values in the cost functions can be handled by introducing auxiliary variables and constraints.[18]

C. Velocity and Acceleration Bounds

To ensure that the planned trajectory respects the velocity limits of the vehicle, maximum and minimum speed constraints were included as follows:

$$\begin{aligned}
&\forall i \in [0 \ldots T-1],\ \forall k \in [1 \ldots K]: \\
&\quad \dot{x}_i \sin(2\pi k/K) + \dot{y}_i \cos(2\pi k/K) \le v_{\max} \\
&\quad \dot{x}_i \sin(2\pi k/K) + \dot{y}_i \cos(2\pi k/K) \ge v_{\min} - M c_{ik} \\
&\quad \sum_{k=1}^{K} c_{ik} \le K - 1, \qquad c_{ik} \in \{0, 1\}
\end{aligned} \tag{8}$$

For the reference velocity model, these constraints were formulated in terms of the input commands $\dot{x}_{\text{cmd},i}$ and $\dot{y}_{\text{cmd},i}$ instead. In the preceding, the maximum speed bound is approximated by constraining the inertial velocity vector to lie inside a K-sided polygon. Using a sufficiently large number M and binary variables c_{ik}, the minimum speed requirement is captured by constraining the velocity vector to lie outside a (smaller) K-sided polygon. The demonstration system 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), their values were accordingly adapted online.

Both the reference velocity model (4) and double integrator model (5) assume homogeneous dynamics in the x and y coordinates, thus ignoring differences in lateral and longitudinal dynamics. This was corrected for by adding linear constraints that capture limits on turn rate and on forward and lateral acceleration. When flying at a relatively constant speed, the following acceleration constraints are sufficient[19]:

$$\begin{aligned}
&\forall i \in [0 \ldots T-1],\ \forall k \in [1 \ldots K]: \\
&\quad \ddot{x}_i \sin(2\pi k/K) + \ddot{y}_i \cos(2\pi k/K) \le a_{\text{lat}}
\end{aligned} \tag{9}$$

for the double integrator model, and

$$\begin{aligned}
&\forall i \in [1 \ldots T-1],\ \forall k \in [1 \ldots K]: \\
&\quad (\dot{x}_{\text{cmd},i} - \dot{x}_{\text{cmd},i-1}) \sin(2\pi k/K) \\
&\quad + (\dot{x}_{\text{cmd},i} - \dot{x}_{\text{cmd},i-1}) \cos(2\pi k/K) \le a_{\text{lat}} \Delta t
\end{aligned} \tag{10}$$

for the reference velocity model. The lateral acceleration bound was set at $a_{\text{lat}} = 18.1$ ft^2/s, corresponding to a maximum turn rate of $\omega_{\max} = a_{\text{lat}}/v_{\min} = 2.6$ 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 $a_{\text{fwd}} = 5.0$ ft^2/s. Therefore, to distinguish between forward and lateral acceleration, the following constraints were added:

$$\begin{aligned}
&\forall i \in [0 \ldots T-1],\ \forall k \in [1 \ldots K]: \\
&\quad \left(\ddot{x}_i + \alpha v_0^{-1} \dot{y}_i\right) \sin(2\pi k/K) + \left(\ddot{y}_i - \alpha v_0^{-1} \dot{x}_i\right) \\
&\quad \times \cos(2\pi k/K) \le \beta a_{\text{lat}} \\
&\left(\ddot{x}_i - \alpha v_0^{-1} \dot{y}_i\right) \sin(2\pi k/K) + \left(\ddot{y}_i + \alpha v_0^{-1} \dot{x}_i\right) \cos(2\pi k/K) \le \beta a_{\text{lat}}
\end{aligned} \tag{11}$$

for the double integrator model, and

$$\begin{aligned}
&\forall i \in [1 \ldots T-1],\ \forall k \in [1 \ldots K]: \\
&\quad \left(\dot{x}_{\text{cmd},i} - \dot{x}_{\text{cmd},i-1} + \alpha v_0^{-1} \dot{y}_i \Delta t\right) \sin(2\pi k/K) \\
&\quad + \left(\dot{y}_{\text{cmd},i} - \dot{y}_{\text{cmd},i-1} - \alpha v_0^{-1} \dot{x}_i \Delta t\right) \cos(2\pi k/K) \le \beta a_{\text{lat}} \Delta t \\
&\left(\dot{x}_{\text{cmd},i} - \dot{x}_{\text{cmd},i-1} - \alpha v_0^{-1} \dot{y}_i \Delta t\right) \sin(2\pi k/K) \\
&\quad + \left(\dot{y}_{\text{cmd},i} - \dot{y}_{\text{cmd},i-1} + \alpha v_0^{-1} \dot{x}_i \Delta t\right) \cos(2\pi k/K) \le \beta a_{\text{lat}} \Delta t
\end{aligned} \tag{12}$$

for the reference velocity model. Here, v_0 is the current absolute ground speed, $\alpha = (a_{\text{lat}}^2 - a_{\text{fwd}}^2)/(2a_{\text{fwd}}) = 30.4$ fps^2/s, and $\beta = \sqrt{(\alpha^2 + a_{\text{lat}}^2)} = 35.4$ fps^2/s. These inequalities describe the intersection of two circles in which the inertial acceleration vector must lie. The short axis of this intersection has length $2a_{\text{fwd}}$ and is aligned with the velocity vector at the first time step. The long axis captures the larger lateral acceleration bound and has length $2a_{\text{lat}}$. As such, 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 Ref. 20.

D. Obstacle Avoidance

The demonstration only considered rectangular no-fly zones aligned with the east-north coordinate frame, although more general polytopes can easily be included in the formulation. As discussed earlier, only the zones lying inside the current detection region $\mathcal{D}_0$ of the aircraft must be accounted for. Denoting these obstacles by $j = 1, \ldots, J$ and specifying 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[15]

$$\begin{aligned}
&\forall i \in [1, \ldots, T],\ \forall j \in [1, \ldots, J]: \\
&\quad x_i \le x_{\min,j} + M b_{ij1}, \qquad -x_i \le -x_{\max,j} + M b_{ij2} \\
&\quad y_i \le y_{\min,j} + M b_{ij3}, \qquad -y_i \le -y_{\max,j} + M b_{ij4} \\
&\quad \sum_{r=1}^{4} b_{ijr} \le 3, \qquad b_{ijr} \in \{0, 1\}
\end{aligned} \tag{13}$$

Using a sufficiently large number M again, the last constraint ensures that at least one of the coordinate inequalities for each rectangle $j = 1 \ldots J$ is active, thereby guaranteeing that the trajectory point (x_i, y_i) lies outside all obstacles. Because the resulting trajectory consists of discrete waypoints, to prevent the UAV from cutting corners the actual obstacles were enlarged with a safety boundary of $d_{\text{safe}} = v_{\max} \Delta t/\sqrt{2} \approx 3200$ ft.

E. Loiter Constraints

As mentioned before, safety can be guaranteed by ensuring that either a left or right loiter circle that lies inside the detection region does not intersect with any of the obstacles $j = 1 \ldots J$. As detailed in Ref. 8, sample points along both circles can be expressed as affine functions of the last state $\boldsymbol{x}_T$ in the planning horizon, for which avoidance constraints similar to Eq. (14) can then be introduced.

As such, $\boldsymbol{x}_T$ is constrained to be an ingress state to a safe loiter pattern.

Using an index l to indicate the N sample points along the circles and a binary variable d to select either the right or left one, the safe loiter constraints at each receding-horizon iteration become[8]

$\forall l \in [1 \ldots N],\ \forall j \in [1 \ldots J]$:

$$\begin{cases} x_T - \alpha_c(\cos l\theta_s - 1)\dot{y}_T - \alpha_c(\sin l\theta_s)\dot{x}_T \\ \qquad \le x_{\min,j} + M\hat{b}_{lj1} + Md \\ -x_T + \alpha_c(\cos l\theta_s - 1)\dot{y}_T + \alpha_c(\sin l\theta_s)\dot{x}_T \\ \qquad \le -x_{\max,j} + M\hat{b}_{lj2} + Md \\ y_T - \alpha_c(\sin l\theta_s)\dot{y}_T + \alpha_c(\cos l\theta_s - 1)\dot{x}_T \\ \qquad \le y_{\min,j} + M\hat{b}_{lj3} + Md \\ -y_T + \alpha_c(\sin l\theta_s)\dot{y}_T - \alpha_c(\cos l\theta_s - 1)\dot{x}_T \\ \qquad \le -y_{\max,j} + M\hat{b}_{lj4} + Md \end{cases} \tag{14}$$

$$\begin{cases} x_T + \alpha_c(\cos l\theta_s - 1)\dot{y}_T + \alpha_c(\sin l\theta_s)\dot{x}_T \\ \qquad \le x_{\min,j} + M\hat{b}_{lj1} + M(1-d) \\ -x_T - \alpha_c(\cos l\theta_s - 1)\dot{y}_T - \alpha_c(\sin l\theta_s)\dot{x}_T \\ \qquad \le -x_{\max,j} + M\hat{b}_{lj2} + M(1-d) \\ y_T + \alpha_c(\sin l\theta_s)\dot{y}_T - \alpha_c(\cos l\theta_s - 1)\dot{x}_T \\ \qquad \le y_{\min,j} + M\hat{b}_{lj3} + M(1-d) \\ -y_T - \alpha_c(\sin l\theta_s)\dot{y}_T + \alpha_c(\cos l\theta_s - 1)\dot{x}_T \\ \qquad \le -y_{\max,j} + M\hat{b}_{lj4} + M(1-d) \end{cases} \tag{15}$$

$$\begin{cases} \sum_{r=1}^{4} \hat{b}_{ljr} \le 3 \\ \hat{b}_{ljr},\ d \in \{0, 1\} \end{cases} \tag{16}$$

Here $\theta_s = 2\pi/N$ is the discretization angle around the circle, and α_c is a (conservative) constant scaling the radius with the ingress velocity. Although this scaling is a linear overapproximation of the actual quadratic dependence, it gives the aircraft an extra degree of freedom when fitting the loiter circles in the obstacle-free areas of the known environment.[8] For example, if necessary, slowing down will enable it to form a tighter 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 miles, the demonstration used $N = 8$ and $d_{\text{loiter}} = 0.6$ miles.

VII. Trajectory Generation Module

The trajectory generation module was implemented in C++ and ILOG's Concert Technologies. To interface with the UAV avionics and guarantee hard real-time execution, it was integrated with Boeing's Open Control Platform.[11] 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 predefined command variables. Through the OCP interface the laptop receives global positioning system (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 subroutines: a preprocessing step, an optimization step, and a postprocessing step, which are now discussed in more detail.

A. Preprocessing

The preprocessing routine is called every second and determines all parameters of the MILP problem. It subsequently 1) selects the correct LTI model, 2) estimates the initial state for the current planning horizon, 3) determines the relevant obstacles, 4) enlarges the obstacles with the appropriate safety band, 5) determines the intermediate waypoint, and 6) 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. The ground velocity of the aircraft is scaled to kilometers/second accordingly.

B. Optimization

The optimization step was 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. In our mission software, this limit was set to 0.85 s. After the allocated time has passed, CPLEX either returns a feasible solution within a predefined optimality gap (set to 10^{-4}), a feasible solution outside the optimality gap, or no solution at all. The last situation occurs when 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 preprocessing 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 remaining time steps and loiter circle of the previous plan.

If the control authority used in the MILP problem (i.e., the admissible acceleration and turn rate limit) is somewhat conservative with respect to 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.[21] Our code therefore uses maximum acceleration and turn rate bounds that are smaller than the actual ones available to the waypoint controller. As a result, infeasibilities of more than two time steps never occurred.

Although the preprocessing step is repeated every second, the optimization function is nominally only executed every 10 s: the ~10-s time constant of the T-33 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), is the optimization routine 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 components.

C. Postprocessing

The postprocessing 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 waypoint counter accordingly. In the nominal case in which a feasible solution is found, the variables of interest are the six new states of the planning horizon (i.e., the new 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 feet per second. Next, the old plan is flushed and replaced by the new one. The backup plan counter is set to one, pointing to the first entry in the waypoint/state list, which 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 that case, the backup plan counter is increased by one to point to the next waypoint of

the existing plan. If the counter exceeds six, 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 to the UAV. Its value is set to the maximum available turn rate at the current velocity, for example, 3 deg/s at 400 fps, which is slightly more aggressive than the maximum 2.6 deg/s accounted for in the planning problem. The turn command thus results in a smaller loiter circle than planned, which introduces some robustness to perturbations along the trajectory. As long as the vehicle remains in the backup plan mode, the MILP optimization is executed every second (but the counter only updated every 10 s) until a new feasible plan is found.

VIII. Simulation and Flight-Test Results

Using the narrative outlined in Sec. II, a variety of sample scenarios was designed, which is depicted in Fig. 3. The flight area is approximately 40 miles across (east to west along the northern boundary) and 30 miles wide (north to south along the western boundary). There are two predetermined no-fly zones (listed as "NFZ 1" and "NFZ 2") and three potential pop-up threats (denoted by "PObs 1," "PObs 2," and "PObs 3"), which can be activated during flight. 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 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.

A. Simulation Results

To aid in the development of our guidance system, a real-time simulation-in-the-loop (SIL) test platform was built at MIT. Besides the OCP, it included Boeing's DemoSim vehicle simulations for the UAV (T-33) and FW (F-15) aircraft, which were executed on separate computers with a Link 16 communication interface. The mission system software ran on two laptops similar to the ones mounted in the aircraft during the test flight experiments. Using wireless ethernet connections through the laboratory LAN, command latency and other real-time issues could be simulated. Beside communications link latency, test conditions included message drop-outs, invalid experiment key selection, data scaling issues, and modeling errors.

Figure 4 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.

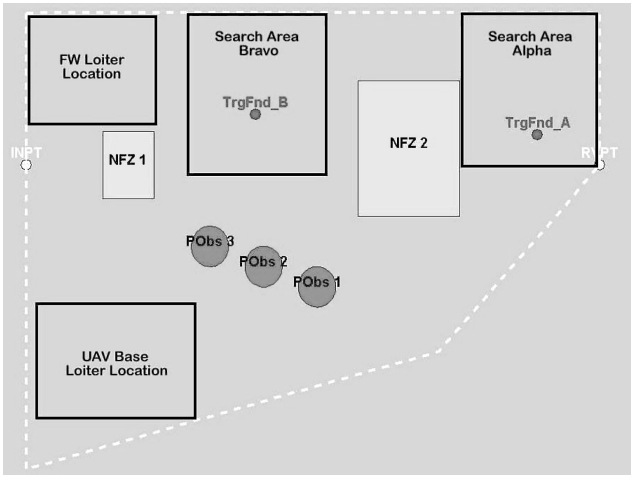

Fig. 3 Sample scenario map for the MIT SEC Capstone Demonstration. The flight area is approximately 40 miles long along the northern boundary.

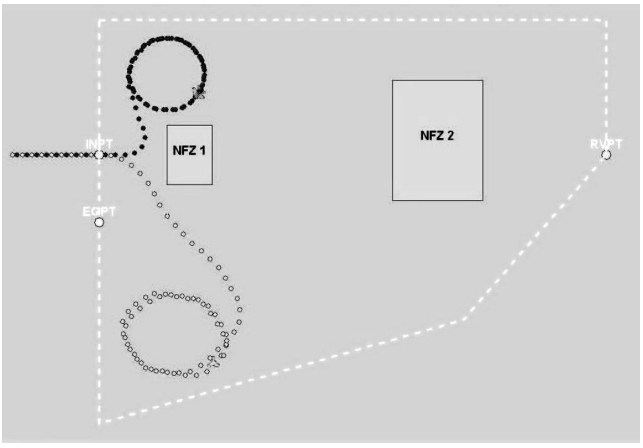

Fig. 4 SIL test 1—Initialization of the SEC demonstration: the UAV (in light) enters a loiter pattern.

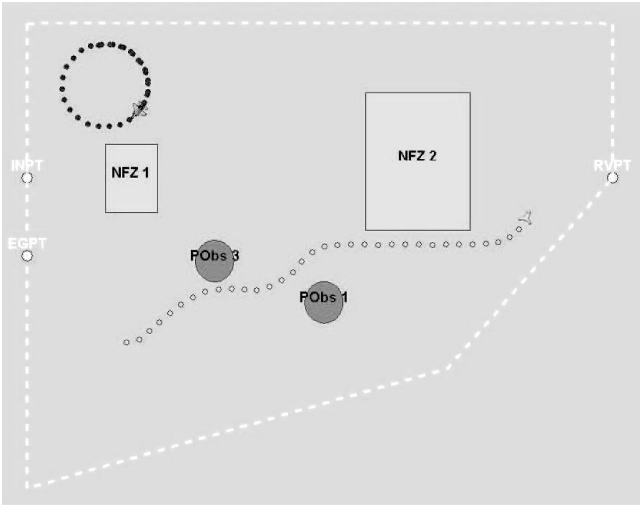

Fig. 5 SIL test 2—Pop-up obstacle test: the UAV safely avoids both pop-up threats.

Next, Fig. 5 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 (PObs 3), the second one (PObs 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 Search Area Alpha to perform a search task.

Figure 6 depicts a test where the UAV was commanded to fly two consecutive missions from the UAV Base Loiter Location. The main objective of this test was to ensure that the vehicle returns to its loiter location after it finishes a certain task (provided that another task was not given). First, the FW WSO commands the UAV to proceed to one of the search areas, but does not issue the Return-To-Base (RTB) command during the mission. Still, after the UAV finishes its search of the task area, it informs the FW WSO that it has completed the search task and will proceed back to the Base Loiter Location

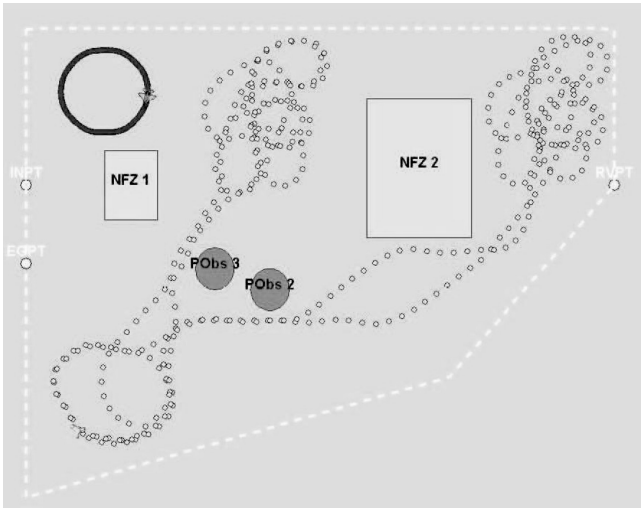

Fig. 6 SIL test 3—Simulated flight with two consecutive search missions.

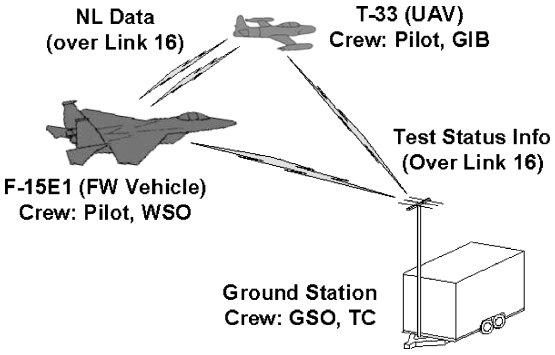

Fig. 7 Flight experiment system-level diagram.

to await another set of commands. This test shows that the software provides the vehicle operators and test directors with flexibility in task and mission management during flight.

Figure 6 also 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 a pop-up obstacle en route to each search area.

B. Flight-Test Results

From mid-April 2004 to mid-June 2004, the final demonstration software was turned over to Boeing Phantom Works in St. Louis for verification and validation testing on a hardware-in-the-loop simulator. After successfully completing this step, the software was transitioned to the actual vehicle setup for testing at NASA Dryden in late June 2004. Figure 7 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 the demonstration 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 predefined list of options, the actual mission was not preplanned. The T-33's rear-seat operator (nicknamed 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. Although the possible locations of the mission task areas and pop-up obstacles were predefined, they were selected randomly in real time, thus introducing another degree of uncertainty into the flight experiment. The test coordinator (TC) monitored the demonstration from the ground station and communicated status information about the local airspace to the pilots.

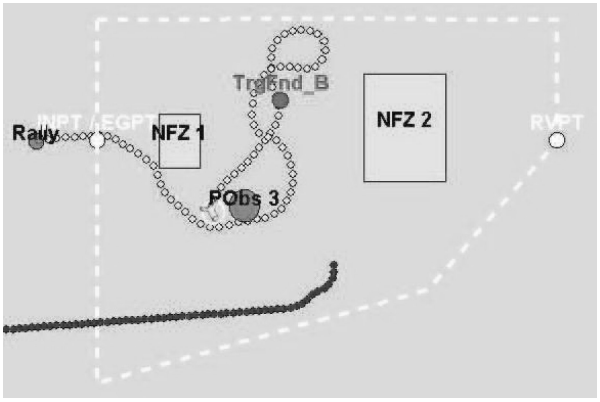

Fig. 8 SEC flight test: UAV/T-33 (in light) with simulated F-15 (in dark).

1. *Flight Test with T-33/UAV and Simulated F-15*

In the first test, depicted in Fig. 8, the T-33/UAV flew a mission with a simulated F-15, whereby the GSO issued the natural language commands. The flight took place in the morning of Thursday, 17 June 2004, with a wind of 5 to 10 kn blowing from the southwest corner of the flight area (at a heading of 220 to 240 deg).

The UAV started west of the ingress point (labelled "INPT/EGPT" in Fig. 8) with a heading of 090. After the GIB initialized the mission software, the UAV began turning south to avoid NFZ 1. After it passed the lower left-hand corner of NFZ 1, the TC notified the T-33/UAV operators that the southwestern corner of the test area was to be avoided because of unplanned flight activity there. As such, when the vehicle was approximately two miles SSW of NFZ 1, the GSO commanded the UAV to proceed to Task Area Bravo before reaching the UAV Base Loiter Location. The UAV responded and began turning left toward it. This verified the flexibility of the mission software and the ability of an operator to easily change the tasks in real time.

Within 2 min of the last 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, even though the obstacle was inserted inside the vehicle's detection region. After passing it, the UAV proceeded to turn north toward Task Area Bravo, initiated the search pattern, and notified the GSO of its status. 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, after which the GSO commanded it to return to base. The vehicle began to turn right, safely flew southwest around pop-up obstacle 3, and sent a "Two Minutes to Task Location" notification. Because the southwest airspace was off limits, however, the UAV was not permitted to fly to its Base Loiter Location, and the mission was ended.

The flight test marked the first time a MILP-based onboard guidance system operating in real time was used to control a UAV. The natural language software effectively communicated mission status to the GSO and was successfully used to command the UAV to perform and change tasks during the flight. The GSO remarked that he found the interface easy to use and helpful in following the progress of the mission.

2. *Flight Test with T-33/UAV and Actual F-15*

In a second test, shown in Fig. 9, the T-33/UAV flew a successful mission with the F-15 WSO issuing the natural language commands.

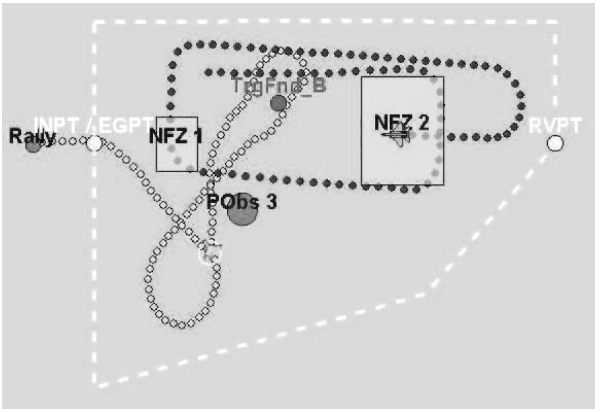

Fig. 9 SEC flight test: UAV/T-33 (in light) and actual F-15 (in dark) providing natural language commands.

This flight took place in the afternoon on Wednesday, 23 June 2004, with a SW wind (220–240 deg) between 10 and 15 kn.

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 Loiter Location. When it had a heading of 270, the F-15 WSO commanded the UAV to fly to Task Area Bravo. It responded and began turning northward. When the GSO inserted pop-up obstacle 3, the UAV notified the F-15 that it detected the threat and successfully avoided the obstacle. As it approached the task area, the UAV informed the F-15 WSO that it was 2 min from the ingress point that was initially given. At that time, the WSO commanded the UAV to change the entrance location. The UAV responded and proceeded to the new ingress point. This again verified the flexibility of the task scheduling and trajectory planning software.

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. The F-15 WSO then commanded the UAV to return to base. The UAV acknowledged the command and flew back to its loiter location, 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 Base Loiter Location.

This flight test marked the first time that a natural language interface was used by a manned vehicle to task and command a UAV in real time. The F-15 WSO also remarked that he found the interface easy to use and helpful in following the progress of the mission. In addition, the test marked the first time that an onboard MILP-based guidance system was used to control a UAV in coordination with a manned vehicle. Furthermore, it was the first successful in-flight cooperation demonstration between the F-15 and UAV for the Boeing team. Overall, both flight tests provided an important proof of concept of the capabilities of the MIT mission software as previously witnessed in the laboratory: the flexibility of the task scheduling software allowed the test team to make adjustments in real time, while the trajectory generation algorithm safely guided the UAV through the environment.

IX. Conclusions

This paper described the development, architecture, and testing of a manned vehicle/UAV mission system that allows an operator in a manned aircraft to issue mission-level commands to an autonomous aircraft in real time. A natural language interface that allows the manned and unmanned vehicle to communicate in languages understood by both agents was presented. A task scheduler then transforms these commands into a dynamic mission plan consisting of task waypoints. The latter are given to a guidance system based on mixed-integer linear programming, which generates safe trajectories through a partially known environment in real time.

The complete mission system was successfully tested using high-fidelity software and hardware-in-the-loop simulations. Actual flight-test results with an F-15 and an autonomous T-33 were presented, which provide an important proof of concept of the benefits and real-time capabilities of the natural language interface and MILP-based guidance methodology. After these first validation steps, the approach and algorithms are ready to be transitioned to larger problems, such as platforms with multiple unmanned vehicles for which safe decentralized trajectory planning and task assignment strategies are being developed. The eventual goal is to have a single operator issue team-level mission commands using natural language. We believe that in the future natural language interfaces will be the most efficient way to communicate with unmanned vehicle systems.

Acknowledgments

The core natural language processing functions were developed together with the Teragram Corporation. The authors would also like to thank James Paunicka, Timothy Espey, Brian Mendel, and Jared Rosson from Boeing Phantom Works for their support with integrating the Massachusetts Institute of Technology (MIT) software in the OCP system, and Yoshiaki Kuwata from MIT for his help with implementing the mixed-integer linear programming module in Concert Technologies. The research was funded by Defense Advanced Research Projects Agency Software Enabled Control contract F33615-01-C-1850.

References

[1]Office of the Secretary of Defense for Acquisition, Technology, and Logistics, "Unmanned Aerial Vehicle Roadmap 2002-2007," Rept. A809414, Dept. of Defense, Washington, DC, Dec. 2002.

[2]Wohletz, J., Castanon, D., and Curry, M., "Closed-Loop Control for Joint Air Operations," *Proceedings of the 2001 American Control Conference*, Vol. 6, IEEE Publications, Piscataway, NJ, 2001, pp. 4699–4704.

[3]Kott, A., *Advanced Technology Concepts for Command and Control*, Xlibris Corp., Philadelphia, PA, 2004.

[4]Samad, T., and Balas, G., *Software-Enabled Control: Information Technology for Dynamical Systems*, Wiley/IEEE Press, Hoboken, NJ, 2003.

[5]Butenko, S., Murphey, R., and Pardalos, P., *Recent Developments in Cooperative Control and Optimization*, Kluwer Academic, Norwell, MA, 2003.

[6]Grundel, D., Murphey, R., and Pardalos, P., *Theory and Algorithms for Cooperative Systems*, Vol. 4, Series on Computers and Operations Research, World Scientific, Hackensack, NJ, 2004.

[7]Floudas, C. A., *Nonlinear and Mixed-Integer Programming—Fundamentals and Applications*, Oxford Univ. Press, Oxford, England, UK, 1995.

[8]Schouwenaars, T., How, J., and Feron, E., "Receding Horizon Path Planning with Implicit Safety Guarantees," *Proceedings of the 2004 American Control Conference*, Vol. 6, IEEE Publications, Piscataway, NJ, 2004, pp. 5576–5581.

[9]*ILOG CPLEX 9.0 User's Guide*, ILOG, Mountain View, CA, 2003.

[10]Mettler, B., Valenti, M., Schouwenaars, T., Kuwata, Y., How, J., Paunicka, J., and Feron, E., "Autonomous UAV Guidance Build-Up: Flight-Test Demonstration and Evaluation Plan," AIAA Paper 2003-5744, 2003.

[11]Paunicka, J., Mendel, B., and Corman, D., "The OCP—An Open Middleware Solution for Embedded Systems," *Proceedings of the 2001 American Control Conference*, Vol. 5, IEEE Publications, Piscataway, NJ, 2001, pp. 3445–3450.

[12]Roche, E., "Parsing with Finite-State Transducers," *Finite-State Language Processing*, edited by E. Roche and Y. Schabes, MIT Press, Cambridge, MA, 1997.

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

[14]Bemporad, A., and Morari, M., "Control of Systems Integrating Logic, Dynamics, and Constraints, *Automatica*, Vol. 35, No. 3, 1999, pp. 407–427.

[15]Schouwenaars, T., DeMoor, B., Feron, E., and How, J., "Mixed Integer Programming for Multi-Vehicle Path Planning," *Proceedings of the 2001 European Control Conference*, European Union Control Association, Saint-Martin d'Hères, France, 2001, pp. 2603–2608.

[16]Bellingham, J., Richards, A., and How, J., "Receding Horizon Control of Autonomous Aerial Vehicles," *Proceedings of the 2002 American Control Conference*, Vol. 5, IEEE Publications, Piscataway, NJ, 2002, pp. 3741–3746.

[17]Kuwata, Y., and How, J., "Stable Trajectory Design for Highly Constrained Environments Using Receding Horizon Control," *Proceedings of the 2004 American Control Conference*, Vol. 1, IEEE Publications, Piscataway, NJ, 2004, pp. 902–907.

[18]Bertsimas, D., and Tsitsiklis, J. N., *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 1997, pp. 17–19.

[19]Richards, A., and How, J., "Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming," *Proceedings of the 2002 American Control Conference*, Vol. 3, IEEE Publications, Piscataway, NJ, 2002, pp. 1936–1941.

[20]Schouwenaars, T., Mettler, B., Feron, E., and How, J.,"Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles," *AIAA Journal of Aerospace Computing, Information, and Communication*, Vol. 1, No. 12, 2004, pp. 629–651.

[21]Kuwata, Y., Schouwenaars, T., Richards, A., and How, J., "Robust Constrained Receding Horizon Control for Trajectory Planning," AIAA Paper 2005-6079, Aug. 2005.