JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 31, No. 6, November–December 2008

Flight Testing a Real-Time Direct Collocation Path Planner

Brian R. Geiger,* Joseph F. Horn,† Gregory L. Sinsley,‡ James A. Ross,* Lyle N. Long,§ and Albert F. Niessner¶
Pennsylvania State University, University Park, Pennsylvania 16802

DOI: 10.2514/1.36533

A path-planning algorithm using direct collocation with nonlinear programming is demonstrated in both simulation and flight tests. Direct collocation, which approximates the states and controls with piecewise polynomials, has been widely applied in space vehicles and manned aircraft, but has only seen limited use in unmanned aerial vehicle applications. The algorithm is successfully used to generate a path that produces maximal surveillance time of a moving or stationary ground target by a sensor mounted on an unmanned aerial vehicle while compensating for aircraft performance or mission constraints. Flight tests of the path-planning algorithm operating in real time onboard an unmanned aerial vehicle are also presented. These tests include surveilling a stationary and moving target with a video camera while compensating for any wind effects. Additionally, the effect of the use of road data in planning the path is simulated by tracking a second unmanned aerial vehicle flying a predefined pattern.

Nomenclature

C_m	=	camera axes to unmanned aerial vehicle axes transformation matrix
$I_{x,y}$	=	image dimensions in pixels
K	=	camera intrinsic-properties matrix
n	=	number of unmanned aerial vehicles
$\boldsymbol{P}$	=	position of unmanned aerial vehicles relative to the target
$\boldsymbol{P_l}$	=	pixel coordinates
$\boldsymbol{P_m}$	=	overall parameter vector
$\boldsymbol{P_s}$	=	state vector for n unmanned aerial vehicles
$\boldsymbol{p_c}$	=	control vector for n unmanned aerial vehicles
q	=	number of targets
R	=	unmanned aerial vehicle direction cosine matrix
r	=	number of nodes
s	=	position of a target on a road, m
T_h	=	horizon time, s
$\boldsymbol{u}$	=	unmanned aerial vehicle control vector
$\boldsymbol{u_a}$	=	approximated control vector
u_l	=	longitudinal acceleration command, m/s^2
u_ϕ	=	bank-angle command, rad
V_t	=	true airspeed, m/s
x, x_t	=	north position of unmanned aerial vehicle and target, m
$\boldsymbol{x_a}$	=	approximated state vector
$\boldsymbol{x_c}$	=	state vector at a collocation point
$\boldsymbol{x_t}$	=	target state vector
$\boldsymbol{x_u}$	=	unmanned aerial vehicle state vector
y, y_t	=	east position of unmanned aerial vehicle and target, m
$\boldsymbol{\Delta}$	=	defect vector
τ	=	segment length, s
ψ	=	heading, rad

Presented as Paper 6654 at the AIAA Guidance, Navigation, and Control Conference, Hilton Head, SC, 20–23 August 2007; received 8 January 2008; revision received 13 May 2008; accepted for publication 16 May 2008.

*Graduate Research Assistant, Department of Aerospace Engineering. Student Member AIAA.
†Associate Professor, Department of Aerospace Engineering. Associate Fellow AIAA.
‡Graduate Research Assistant, Department of Electrical Engineering.
§Distinguished Professor, Department of Aerospace Engineering. Fellow AIAA.
¶Senior Research Associate, Emeritus, Guidance Systems Technology, Applied Research Laboratory.

I. Introduction

AS THE roles of unmanned aerial vehicles (UAVs) in military and civilian use are expanded, the need arises for increased autonomy. An integral step in increasing UAV autonomy is the ability for automatic path planning considering external events or situations. The ability to simply command a UAV to observe a target and have the UAV plan the best flight path given sensor characteristics, obstructions, target motion, multiple targets, or multiple UAVs would be useful. We propose an implementation for a path planner using direct collocation with nonlinear programming and present flight-test results. The direct collocation method provides an easy way to account for both the performance limits of an unmanned aircraft and the characteristics of an onboard sensor.

Focusing on the observation task at hand, Rysdyk [1] examines the effect of wind on a UAV observing a ground target using a gimbaled camera. He shows that it is possible for the aircraft to maintain a constant line of sight to the target in the aircraft body frame, such as aiming a wingtip at a target. The implementation we preset uses a downward-facing camera, however, the method could accommodate changes in camera orientation. Frew et al. [2] use a group of fixed-wing aircraft to follow a road using visual information or to track ground targets by uploading Global Positioning System (GPS) data. Circular and sinusoidal paths are used to maintain a set distance if targets move too slowly for the UAVs. The algorithm was successfully flight-tested. Dobrokhodov et al. [3] also present research with similar goals. They use a small gimbaled camera to track and estimate position coordinates of a moving target. The target-tracking algorithm was designed to be robust in the presence of target-tracking loss.

Direct collocation with nonlinear programming (DCNLP) is based on research by Hargraves and Paris [4] on a low-Earth-orbit booster problem and a supersonic aircraft time-to-climb algorithm. The DCNLP method was first introduced by Dickmanns and Well [5] as a general method for solving nonlinear optimal control problems. The method has seen wide use in spacecraft and satellite research. The problem of low-thrust spacecraft trajectories is investigated in [6–9]. In particular, Herman and Conway [6] use higher-order Gauss–Lobatto quadrature rules instead of the original trapezoidal and Simpson rules. This change allows for increased accuracy with a reduced number of subintervals. The number of nonlinear programming parameters is smaller as a result. Rendezvouses between two power-limited spacecraft of different efficiencies are studied by Coverstone-Carroll and Prussing [7], showing that DCNLP can be applied to more than one vehicle. Several authors

[8,10] note that no a priori assumptions about the optimal control solution are required for DCNLP to converge to a solution. In [9], DCNLP is identified to be in a general class of direct transcription methods. The relationship between the original optimal control problem and the approximation afforded by DCNLP is examined. Horie and Conway [11] study optimal aeroassisted orbital intercept using DCNLP. They note that the direct method allowed for easy inclusion of the complicated heating limit constraints required for this problem, compared with the two-point boundary-value-problem formulation, which is an indirect method. Additionally, they find that DCNLP has an advantage over the two-point boundary-value-problem formulation in terms of problem size and robustness. Direct collocation has also seen use in atmospheric applications. Notably, aircraft conflict resolution in three dimensions for multiple aircraft is investigated in [12]. Betts and Cramer [13] use direct collocation for a commercial aircraft trajectory optimization subject to path constraints imposed by the Federal Aviation Administration. The authors note that the method allows for a natural implementation of these constraints and is robust while rapidly producing a solution.

Although direct collocation has been widely used in space applications and more moderately for full-sized aircraft, it has seen limited use in UAV applications. It has been applied to unmanned glider [14,15] dynamic soaring and powered-UAV [16] dynamic soaring, wherein the aircraft recovers energy from the atmosphere by crossing wind velocity gradients. The authors noted that DCNLP was well suited to solving this problem. Borrelli et al. [17] investigate the method to provide centralized path-planning along with collision avoidance for UAVs. The collision avoidance applies to both other aircraft and ground-based obstacles. Misovec et al. [18] use a collocation method to generate flight paths while considering the radar signature of the UAV. The method is not DCNLP: rather, it uses the NTG [19] package developed at the California Institute of Technology (collocation, but not solved with nonlinear programming).

The main contribution of this research is to apply DCNLP to the problem of tracking moving or stationary targets with one or more UAVs. The objective is to generate, in real time onboard the UAV, a path that provides maximum viewing time for a sensor mounted in the UAV. The target position is assumed to be known at all times. This assumption is made because target recognition and localization is not a trivial problem. In addition, the effect of using known road data in the DCNLP algorithm is examined. Based on the preceding reviewed literature, DCNLP is a good candidate for the work (no specific initial guesses, use of complicated objective and constraints, ability to trade accuracy for computation speed, and general robustness). The research is intended to be flight-tested on the Pennsylvania State University Applied Research Lab/Aerospace Engineering (ARL/PSU) testbed UAV. This UAV is described in the following section.

For use in future work, a computer vision algorithm was developed to locate 75-cm red balls on the ground using the Webcam. The algorithm runs a color filter on the image and then finds the red blobs with the appropriate size and shape. With terrain data obtained from the USGS, a virtual world model of the flight area was used to calculate the position of the red balls on the ground. Because the orientation of the aircraft and Webcam is known, the computer finds the GPS coordinates of the red balls by projecting the line-of-sight vector onto the terrain surface and calculating the point of intersection. This localization algorithm has not yet been used in conjunction with the path planner and is mentioned as a proposed solution to the target localization problem.

II. UAV Testbed Description

The UAV used in this research is a modified Sig Kadet Senior. This aircraft has a payload of about 5 lb, gross weight of 14 lb, and an 80-in. wingspan. The payload includes a CloudCap Piccolo autopilot, a Ampro single-board computer equipped with a 1.4-GHz processor, a universal serial bus (USB) Webcam or a digital camera, and a 4000-mAh lithium polymer battery. The camera is hard-mounted to the airframe and points downward. The Webcam used for the work described in this report is a Logitech Ultra Vision USB Webcam with a 75-deg horizontal and 54-deg vertical field of view. Video output has a resolution of 640 by 480 pixels and is captured using OpenCV [20], an open-source computer vision library created by Intel in C++. The UAV is also able to accept a 4-megapixel Canon digital camera. A picture of the UAV is shown in Fig. 1. The ARL/PSU UAV team has three Piccolo-equipped UAVs, two with onboard single-board computers. More information on the aircraft can be found in [21]. The Sig Kadet platform has proved itself to be a reliable platform for research. Over the summer and fall of 2006 and summer of 2007, the team has logged over 40 flights. Much of the work in the fall of 2006 involved achieving flight with two UAVs simultaneously to support future collaborative research efforts [22].

III. Overview of DCNLP

Direct collocation with nonlinear programming is a method for solving optimal control problems with any type of nonlinear constraints and objective functions. Consider a nonlinear dynamic system with a set of nonlinear equality and/or inequality constraints:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

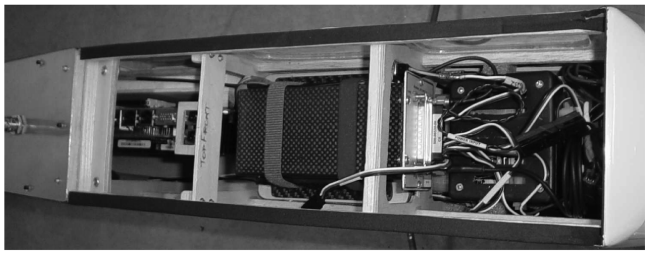
$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}) \leq 0 \tag{2}$$

where $\boldsymbol{x}$ and $\boldsymbol{u}$ are the state and control input vectors. We seek the control input $\boldsymbol{u}(t)$ that minimizes a scalar objective function of the form

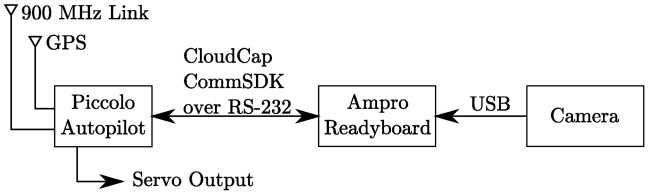
$$J = \int_{t_0}^{t_f} \gamma(\boldsymbol{x}, \boldsymbol{u})\, \mathrm{d}t \tag{3}$$

In general, any number of the state variables can have their initial values x_0 or their final values x_{t_f} specified, and the final time t_f can be fixed or free to vary. In this paper, we consider a problem in which the

a) Exterior view



b) Interior view with piccolo autopilot and computer visible



c) Block diagram of overall system

Fig. 1 ARL/PSU testbed UAV.

initial conditions and final time are specified and the final state vector is free to vary. The DCNLP method provides an approximate numerical solution to this problem by discretizing the state and controls into a set of equally spaced points in time called nodes. A receding-horizon approach is used in conjunction with DCNLP to make the optimization problem tractable for missions of any significant length of time. The optimization is performed over a finite length of time, denoted as the *horizon time* T_h. The optimization is then repeated after a set amount of time has passed, called the horizon update interval. The update uses the latest data available from the aircraft.

To apply DCNLP over a finite time interval, the horizon time is divided into n equal segments of length τ s. The endpoints of these segments are the nodes. The states and controls from the equations of motion are defined at these nodes, and the segments between the nodes are approximated. In this case, we follow the method presented by Hargraves and Paris [4]. Hermite polynomials of order 3 approximate the state trajectories, and controls are represented linearly. Hermite polynomial segments are used for the state interpolation because a segment can be defined completely using the endpoint values and first derivatives at the endpoints. This corresponds to $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, which are given at each node. The polynomial coefficients are given in Eq. (4), where t is nondimensional segment time. Thus, for the ith segment, the approximated states and controls are

$$\begin{aligned}\boldsymbol{x}_{a_i}(t) &= [2(\boldsymbol{x}_i - \boldsymbol{x}_{i+1}) + \dot{\boldsymbol{x}}_i + \dot{\boldsymbol{x}}_{i+1}]t^3 \\ &\quad + [3(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) - 2\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_{i+1}]t^2 + \dot{\boldsymbol{x}}_i t + \boldsymbol{x} \\ \boldsymbol{u}_{a_i}(t) &= \boldsymbol{u}_i + (\boldsymbol{u}_{i+1} - \boldsymbol{u}_i)t\end{aligned} \tag{4}$$

To ensure that the approximating polynomials accurately represent the equations of motion, the derivative of the midpoint of each polynomial segment, $\dot{\boldsymbol{x}}_{a_i}(\frac{1}{2})$, is compared with the equations of motion evaluated at the midpoint, $f(\boldsymbol{x}_{a_i}(\frac{1}{2}), \boldsymbol{u}_{a_i}(\frac{1}{2}))$. This collocation of the approximated and actual derivatives gives the method its name. Carrying out the expansion for the interpolated states at the collocation points (accounting for the nondimensionalized segment time) results in

$$\boldsymbol{x}_{c_i} = (\boldsymbol{x}_i + \boldsymbol{x}_{i+1})/2 + \tau(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_{i+1})/8 \tag{5}$$

Additionally, the slope of the interpolated states at the collocation points are

$$\dot{\boldsymbol{x}}_{c_i} = -3(\boldsymbol{x}_i - \boldsymbol{x}_{i+1})/(2\tau) - (\dot{\boldsymbol{x}}_i + \dot{\boldsymbol{x}}_{i+i})/4 \tag{6}$$

The *defect* is defined as

$$\boldsymbol{\Delta} = f(\boldsymbol{x}_{c_i}, \boldsymbol{u}_{c_i}) - \dot{\boldsymbol{x}}_{c_i} \tag{7}$$

When $\boldsymbol{\Delta}$ is driven toward zero by choosing appropriate values of $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, the approximating polynomials will accurately represent the equations of motion if a cubic polynomial is capable of doing so. Thus, the constraints on the problem are $\boldsymbol{\Delta} = 0$ (additional limits on the states and controls at each node to account for aircraft performance and mission limits), and the states and controls at the first node are held constant.

A. Specific Parameterization

To use a nonlinear solver, the states and controls at each node are collected into a single-column *parameter vector*. The following paragraph details the layout of this vector. Let $\boldsymbol{x}_u$ be the state vector of an aircraft, let $\boldsymbol{x}_t$ be the state vector of a target, and let $\boldsymbol{u}$ be the aircraft control vector. For the simplified dynamic model, the states are the north and east positions of the UAV and target, (x, y, x_t, y_t), airspeed of the UAV, (V_t), and heading of the UAV, (ψ). The controls are longitudinal acceleration command (u_l) and bank-angle command (u_ϕ).

$$\boldsymbol{x}_u = \begin{bmatrix} x \\ y \\ V_t \\ \psi \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} u_l \\ u_\phi \end{bmatrix} \qquad \boldsymbol{x}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} \tag{8}$$

The complete dynamic model is given next for one aircraft and one target. The aircraft equations account for a constant wind speed:

$$\begin{aligned}\dot{x} &= \dot{\boldsymbol{x}}_{u_1} = V_t \cos(\psi) - V_{\text{wind}_N} \\ \dot{y} &= \dot{\boldsymbol{x}}_{u_2} = V_t \sin(\psi) - V_{\text{wind}_E} \\ \dot{V}_t &= \dot{\boldsymbol{x}}_{u_3} = u_l \\ \dot{\psi} &= \dot{\boldsymbol{x}}_{u_4} = g \tan(u_\phi)/V_t \\ \dot{x}_t &= \dot{\boldsymbol{x}}_{t_1} = V_{\text{tgt}_N} \\ \dot{y}_t &= \dot{\boldsymbol{x}}_{t_2} = V_{\text{tgt}_E}\end{aligned} \tag{9}$$

Now let $\boldsymbol{p}_s$ be the vector of all the states of all n UAVs and q targets at a single node, and let $\boldsymbol{p}_c$ be the vector of all n control vectors (for each UAV) at a single node (targets do not have control inputs):

$$\boldsymbol{p}_s = \begin{bmatrix} \boldsymbol{x}_{u0} \\ \boldsymbol{x}_{u1} \\ \vdots \\ \boldsymbol{x}_{un-2} \\ \boldsymbol{x}_{un-1} \\ \boldsymbol{x}_{t0} \\ \boldsymbol{x}_{t1} \\ \vdots \\ \boldsymbol{x}_{tq-2} \\ \boldsymbol{x}_{tq-1} \end{bmatrix} \qquad \boldsymbol{p}_c = \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{n-2} \\ \boldsymbol{u}_{n-1} \end{bmatrix} \tag{10}$$

Now the problem matrix $\boldsymbol{P}_m$ is assembled for r node points:

$$\boldsymbol{P}_m = \begin{bmatrix} \boldsymbol{p}_{s0} & \boldsymbol{p}_{s1} & \cdots & \boldsymbol{p}_{sr-2} & \boldsymbol{p}_{sr-1} \\ \boldsymbol{p}_{c0} & \boldsymbol{p}_{c1} & \cdots & \boldsymbol{p}_{cr-2} & \boldsymbol{p}_{cr-2} \end{bmatrix} \tag{11}$$

The states are organized in the matrix such that each row contains the values of a single (particular) state at each node point. However, the nonlinear solver requires a vector of all the problem variables. Therefore, the problem matrix is reshaped by transposing it and inserting each column sequentially into the parameter (problem) vector. The final parameter vector is not shown, to conserve space.

The complete list of problem constraints specific to the preceding formulation is given in Eq. (12). Note that the initial state and controls $\boldsymbol{x}_0$ and $\boldsymbol{u}_0$ are the state and controls of the UAV and target at the time the optimization is started and remain constant for the optimization. Through the use of constraints on the bank-angle command u_ϕ, the turn rate can be limited. Similarly, longitudinal acceleration is limited by the constraint on u_l:

$$\begin{aligned}\boldsymbol{\Delta} &= 0 \\ \boldsymbol{x}_0 &= \boldsymbol{x}_0 \\ \boldsymbol{u}_0 &= \boldsymbol{u}_0 \\ V_{\min} \le V_{ti} &\le V_{\max} \\ u_{l_{\min}} \le u_{li} &\le u_{l_{\max}} \\ u_{\phi_{\min}} \le u_{\phi_i} &\le u_{\phi_{\max}}\end{aligned} \tag{12}$$

When the path planner is started, an initial guess for the path over the horizon is required. Because DCNLP itself requires no specific initial guess to converge, the current heading and speed of the UAV is used to extrapolate a straight path out to the end of the horizon time.

B. Objective Function

The desired behavior for the UAV is to maximize sensor coverage of the target. The objective function chosen to achieve this behavior

is defined next:

$$J = \int_{t_o}^{t_f} [w_1 u_l^2 + w_2 u_\phi^2 + w_3((x - x_t)^2 + (y - y_t)^2) + w_4 J_{\text{tiv}}] \tag{13}$$

The first two terms penalize control effort (longitudinal acceleration and bank angle), and the third term weights the square of the distance to the target. The fourth term, J_{tiv}, is a target-in-view cost function that has its minimum value when the target is at the center of the image plane of the onboard camera and reaches its maximum value when the target is out of the camera frame. This function is described in more detail next. Note that the third term (distance to target) is required so that in the case that the target is too far away to be viewed by the aircraft, it will attempt to get closer to the target. In the DCLNP solution, the cost function is calculated by performing a numerical integration of Eq. (13) along the state and control trajectory defined by the nodes. The objective function is readily expanded to include cost associated with multiple aircraft by including additional control-effort and distance-to-target terms. The target-in-view cost can be also expanded to include multiple aircraft, as discussed next.

The target-in-view cost is calculated by transforming the target's world coordinates into image plane coordinates (pixel location, $P_l = [\,P_{lx} \quad P_{ly}\,]^T$) using a homography [23] (see Fig. 2). This transformation takes into account the attitude of the UAV, its position relative to the target, the focal length of the camera, the sensor size of the camera, and the orientation of the camera as it is mounted in the airframe. The homography matrix calculation is given in Eq. (14):

$$H = KC_m T = KC_m \begin{bmatrix} R & -RP \\ 0 & 1 \end{bmatrix} \tag{14}$$

where K is the camera intrinsic-properties matrix, C_m is the rotation matrix between camera axes and aircraft axes, R is the direction cosine matrix of the UAV's Euler angles, and P is the position of the UAV relative to the target in world coordinates. The pixel coordinates are then calculated from the world coordinates in Eq. (15):

$$\bar{P}_l = HP \qquad P_l = (1/\bar{P}_{lz})[\,\bar{P}_{lx} \quad \bar{P}_{ly}\,]^T \tag{15}$$

This calculation can be simplified by removing the appropriate Z-coordinate rows because the model uses 2-dimensional motion at a fixed altitude. In this formulation, we account only for the roll attitude and heading of the aircraft, further simplifying the preceding calculation. Pitch attitude is assumed to be zero, but the effect of pitch could readily be added if more detailed equations of motion are used. We also assume a fixed camera looking out from the bottom of the fuselage of the aircraft. However, different camera installation orientations and a gimbaled camera are accounted for by the C_m matrix. The target-in-view cost is zero if the target is in the center of the images and varies parabolically to the edge of the image, where its reaches its maximum value of 1. Outside of the image bounds, the cost function is held constant at one. The target-in-view cost for a single aircraft and a single target is given by Eq. (16):

$$J_{\text{tiv}} = \min\left(\max\left(\frac{4P_{lx}^2}{I_{x_{\max}}^2}, \frac{4P_{ly}^2}{I_{y_{\max}}^2}\right), 1.0\right) \tag{16}$$

where $I_{x_{\max}}$ and $I_{y_{\max}}$ are the size in pixels of the image plane. For multiple aircraft performing surveillance on a single target, the minimum J_{tiv} of all aircraft is used. Thus, the cost is based on the aircraft that has the best view of the target, and it is assumed that there is no particular benefit gained from multiple aircraft simultaneously observing the target. Figure 3 shows a visualization of the target-in-view cost.

As a final overview of the entire process, the problem is parameterized by the method in Sec. III.A. The problem is then solved by searching for the parameter vector [transformed form of Eq. (11)] that minimizes objective function (16) subject to constraints (7) and (12).

IV. Implementation Issues and Initial Flight Tests

Initially, we chose to write a separate path-navigator program. It would read the path output from the optimization and send turn rate commands to the autopilot. The reason for this was to have greater control over how the path is followed. Only limited control is available through the gains on the autopilot. However, the onboard computer must run the path planner, the path navigator, and any image-capture software. In light of the requirement to run the path planner in real time, it would be best to offload as many tasks from the onboard computer as possible. Additionally, there would be increased programming complexity because communications between the path navigator and planner would be required. The autopilot is provided with a software development kit that makes communications with it trivial. Therefore, we chose to convert the path-planner output to a set of waypoints to be sent to the autopilot instead of continuing to work separately on the path-navigator algorithm.

The connection between the path-planning algorithm and the Piccolo autopilot is then fairly simple. Once a path is computed, the x and y coordinates of each node are converted to latitude and longitude. The string of points is converted to the proper format and sent to the Piccolo (using the Cloud Cap Software Development Kit). Because there is a processing lag, after sending the waypoint list, the path planner chooses the correct waypoint on the list to which to command the autopilot to fly. The autopilot will then follow this path until the next update. Speed commands are sent directly by periodically checking where the UAV is on the computed path and sending the appropriate speed command.

A. Integrating External World Data

The initial work for the path-planner algorithm resulted from fairly simple requirements: create a path such that the time spent with the sensor on the target is maximized. However, by expanding the requirements, the planner becomes more useful. One such way to increase capability of the planner is to include external information of the surrounding area (e.g., roads). With the use of known roads, the planner becomes less reliant on observations of target motion to be able to predict future positions. For example, if the path planner has road data for the area it is observing, it may be reasonable to assume

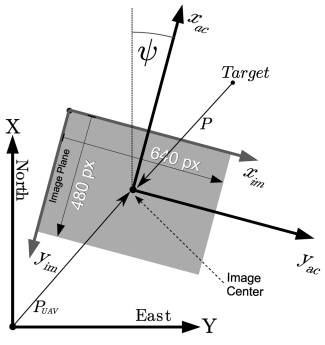


Fig. 2 Axes used in the homography.

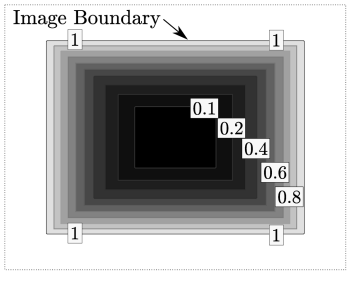


Fig. 3 Plot of the target-in-view objective function.

that a vehicle will follow the road it is on, and therefore, the path planner already knows the path the vehicle will most likely take. With this information, the path planner has access to a more complete model of where a vehicle might go. As shown in Fig. 4, estimates of target speed V and position s along the road are used to make a prediction. Because the vehicle's motion is 1-dimensional along a road, the equation of motion is simply

$$\dot{s} = V \tag{17}$$

The vehicle's location is given by

$$[x_t, y_t] = f(s) \tag{18}$$

Two simulation results using this feature are given in Sec. V.A.

B. Real-Time Implementation

A C++ version of the path-planning algorithm capable of running in real time was developed and is planned to be integrated with the ARL/PSU intelligent controller software [21]. The path planner is being tested as a standalone version before integration. The SNOPT [24] package was selected for use as the nonlinear solver based on recommendations of others who have used it. Because SNOPT has a C++ and MATLAB interface, it can be integrated with both the intelligent controller software and with the existing MATLAB simulation. Verifying the correctness of the C++ implementation compared with the MATLAB implementation is aided by the ability to use the same solver in both languages. The C++ version of the path planner is used in flight test.

A second change was the use of the analytical derivatives of the objective function. Initially, numerical derivatives were used. However, numerical derivatives are slow compared with analytical derivatives, and the path planner is intended to operate online. Of course, analytical derivatives can be very complicated and tedious to obtain. Computer algebra packages such as Mathematica or Maxima can be used to calculate the full derivative, but the resulting expression is very long and most likely contains redundant calculations. If the derivatives of smaller sections of the objective function are taken, the final objective and constraint Jacobian can be obtained using the chain and product rules of differentiation. This results in a longer but more manageable analytical derivative.

Several compromises must be made for the DCNLP method to run in real time. Processing time is essentially driven by the complexity of the objective function. The most complex part of the current objective function is the target-in-view cost, which requires calculation of the pixel coordinates given the real world coordinates of the target and position and orientation of the UAV. Furthermore, an analytical expression for the integral over the collocation points cannot be written, unlike the distance-to-target and control-effort costs. Numerical integration is used, which adds more computation time. Several options present themselves for real-time operation. The first is a reduction in the number of nodes. This would result in fewer calculations at the expense of reduced accuracy in the interpolating polynomials used to approximate the UAV and target equations of motion. In turn, this means that the generated path would have a greater probability to exceed the physical limitations of the UAV (such as turn rate). However, by reducing the horizon time, this effect can be mitigated. A second option is to increase the horizon update interval. This means that the UAV would follow a generated path for a longer amount of time, without the benefit of an update of the world state.

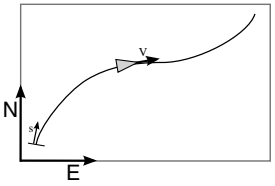


Fig. 4 Path-data illustration.

C. Hardware-in-the-Loop Simulation

Before flight tests were performed with the path planner operating in real time onboard the UAV, hardware-in-the-loop (HIL) simulation was performed with the same computer that is flown on the UAV. Several combinations of horizon time, number of nodes, and horizon update interval were tested to find settings that would be best to flight test. A stationary target was used in the simulation. These HIL simulations test the ability of the path planner to generate a viable path and to do so within the horizon update interval. The standard setup used in simulation was 11 nodes, a 30-s horizon time, and a horizon update every 1.5 s. Using 11 nodes, the algorithm takes around 3 s on average and up to 6 s to generate a new path. This produced a very smooth cloverleaf pattern around a stationary target, however, it was too slow to use in real-time operation because of the very short path update interval. Increasing the update interval to 6 s allowed the path planner to generate a new path before another update was required. Although this was satisfactory, the path generation was still fairly slow at 3 s per path. To obtain a shorter path-generation time, the path planner was tested using 7 nodes and a 20-s horizon time. On average, the path planner requires around 1 s to generate a new path in this configuration. Using an update interval of 4 s with these settings generates a viable path while maintaining a buffer between the time required to generate a path and the path update interval. The two configurations tested in flight tests are given in Table 1. The Results section discusses the advantages and disadvantages of these configurations.

V. Results

The limits shown in Table 2 were used in the following simulation and flight-test results. They are based on the estimated performance limitations of the Sig Kadet Senior. As a safety margin, the bank-angle limit is lower than that of which the UAV is capable.

A. Simulation Results

The path-planning algorithm was implemented initially in MATLAB. The optimal paths were solved in MATLAB using the `fmincon()` function. The MATLAB implementation served as a model for the C++ implementation to be run onboard the UAV. The following are selected results from the simulation.

The first case, shown in Fig. 5, is a single UAV with a stationary target. The resulting trajectory is a cloverleaf pattern. Initially, the UAV accelerates to maximum speed to close the distance to the target and then drops to minimum speed to surveil the target. The cloverleaf pattern develops because of the bottom-pointing camera. The UAV must fly directly over the target to allow the camera to capture a view and then double back to capture another view. The second selected case is a single UAV and stationary target in the presence of a 15-kt steady wind. Figure 6 shows the ground track of the UAV for this case. The generated optimal path is a figure-8 pattern that makes only upwind turns. The UAV flies across the target while drifting downwind before making an upwind turnaround. When the UAV arrives near the target, there is an initial excursion away from the

Table 1 Path-planner configurations used in flight testing

Configuration	Nodes	Horizon length	Update interval
1.	11	30.0 s	6.0 s
2.	7	20.0 s	4.0 s

Table 2 UAV performance constraints

Constraint	Value
Stall speed	$V_{\min} = 22$ kt
Maximum speed	$V_{\max} = 50$ kt
Maximum longitudinal Acceleration	$u_{l_{\max}} = 10$ ft/s^2
	$u_{l_{\min}} = -10$ ft/s^2
Maximum bank angle	$\phi = \pm 30$ deg

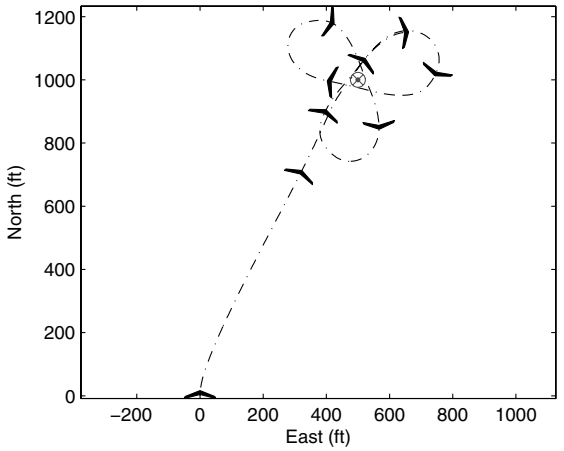


Fig. 5 MATLAB simulation of a single UAV (dash-dot) and a stationary target (circled ×).

target, however, the UAV settles to the relatively repeating figure-8 pattern. The third selected case is the case of two UAVs and a stationary target. Two identical UAVs are equipped with identical cameras. To avoid collisions (in practice), one UAV was simulated at 300-ft altitude and the other was simulated at 400 ft (anticollision methods could be included in the problem formulation). Because the target is stationary, the cloverleaf pattern of the previous calm-wind case develops for both UAVs. However, because the optimization considers both aircraft together, the resulting trajectory staggers the passes over the target so that for a majority of the time, only one UAV has the target in view. This behavior is a result of the way the target-in-view objective function is set up for multiple aircraft. No specific priority is given to one UAV over another, and there is no benefit assumed to having both aircraft view the target simultaneously. Figure 7 shows the paths of the two UAVs, and Fig. 8 shows the distance of each UAV to the target. The bold sections of the lines in Fig. 8 indicate when the target was in view of that particular UAV. Note that the target is alternately covered between aircraft as they pass over. With two UAVs, the target receives sensor coverage almost 100% of the time that the UAVs are on location.

The final selected case is a moving target along a road. The initial MATLAB simulation test of the path planner with road data included simply used a road that makes a 90-deg turn with a 50-ft radius every 300 ft, simulated for 120 s. The path generated is shown in Fig. 9. The solid gray line represents the target's path along the road, and the black dash-dot line is the UAV's path. The field of view of the camera at a selected point is shown by the dashed box. In this case, the speed

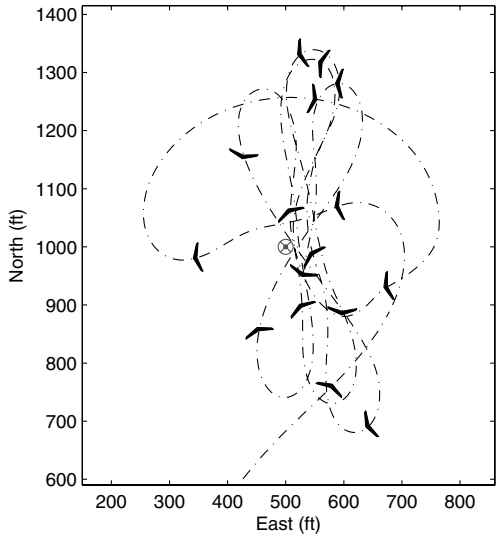


Fig. 6 MATLAB simulation of a single UAV and a stationary target in a 15-kt steady wind from the west (from the left).

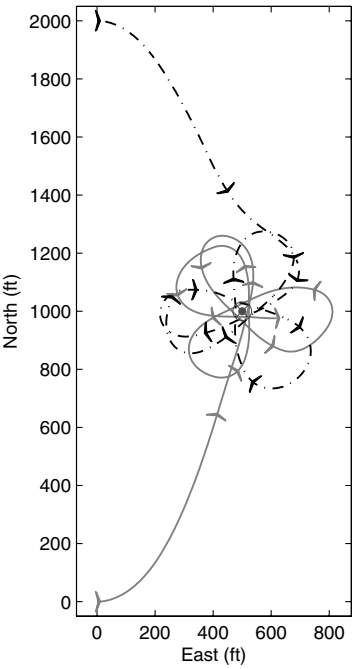


Fig. 7 MATLAB simulation of a UAV pair (dash-dot and gray lines) and a stationary target (circled ×) at (500, 1000).

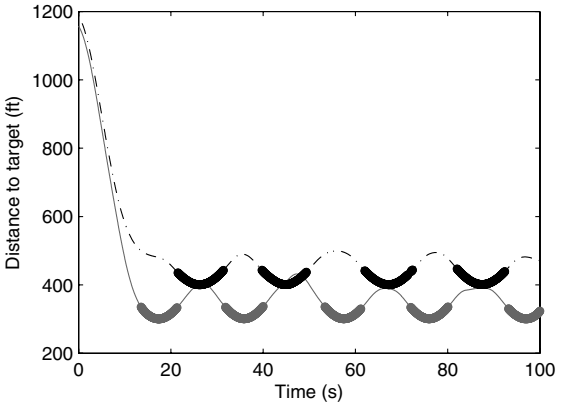


Fig. 8 Distance from UAV to target where bold lines indicate that the target is imaged by the UAV. The offset in distance is due to a difference in altitude of the UAVs.

of the target was within the speed range of the UAV. The target is in view of the UAV's camera at all times. An interesting situation occurs when the vehicle is capable of traveling faster than the UAV, but the vehicle must use a curvy road. The path planner accounts for this by following the target in a "looser" fashion, making up for the speed difference by taking advantage of the curves in the road. Figure 10 illustrates this behavior. For this run, the target speed is set to 100 ft/s and the maximum UAV speed is 87 ft/s. The path planner creates a path that cuts across the curves in the road to make up for the speed difference. Even though the target is moving faster

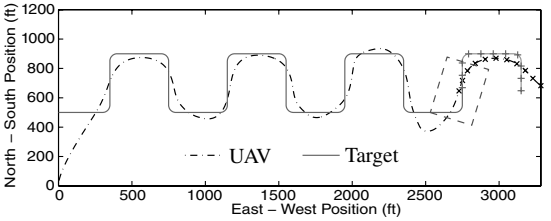


Fig. 9 Path-planner simulation with road data (dashed polygon is the camera field of view).

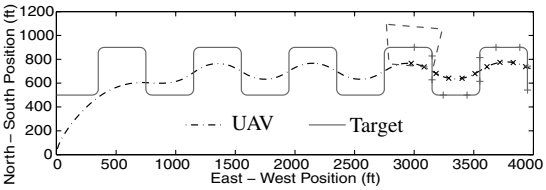


Fig. 10 Path-planner simulation with road data, in which the target is faster than the UAV (dashed polygon is the camera field of view).

than the UAV, the camera is always viewing the target because the UAV banks to keep it in view. With a gimbaled camera, the banking would not be necessary.

Two simulation cases not shown here bear mentioning. The first (trivial) case is when the target is moving in a straight line at a speed that the UAV is able to match. In this case, the UAV simply flies directly over the target. However, this simple case makes it clear that from a coordinate system attached to the target, there is no difference between a moving target and a steady wind for the UAV. The second case considers a target moving slower than the stall speed of the UAV. In this case, the UAV observes the target while it can, breaks off to allow the target to pass it, and then turns back over the target and restarts surveillance.

B. Flight Testing

The path planner was flight-tested while operating in real time onboard the UAV. The test scenarios performed include a stationary target in calm- and steady-wind conditions, a walking person, a moving vehicle, and a second UAV flying at a lower altitude. The results of these tests are discussed subsequently. Figures that show the ground track of the UAV use a grayscale aerial photo of the flying field as the background. The outlined runway is 3100 ft long and 210 ft wide. The end of the ground track is indicated by a triangle. Also visible are 5 dots that mark the locations of barrels used for targets. If they are relevant to the results, they are circled and labeled. For cases with two UAVs, the tracker UAV is shown as a solid line and the target UAV is shown as a dashed line.

1. *Stationary Target*

The airfield at which we fly has barrels lining the runway. These barrels provide convenient targets because they are painted red and can easily been seen by the camera onboard the UAV. One of these barrels was chosen as a stationary target to observe. Its latitude and longitude was input to the path planner before starting the test. Figure 11a shows the ground track of the UAV while under the direction of the path planner. The familiar cloverleaf pattern seen in simulation is readily apparent. Figure 11b is a histogram showing the time required to calculate each new path update. A majority of the path updates require under 1 s, and no path update took longer than the update interval (4 s for this flight), showing that the planner is operating in real time. For both configurations, the camera has a view of the target 41% of the time. This percentage is measured by watching the recorded onboard video after the flight and measuring the length of each time the target is in view and dividing by the length of time from the first view of the target to the end of the test.

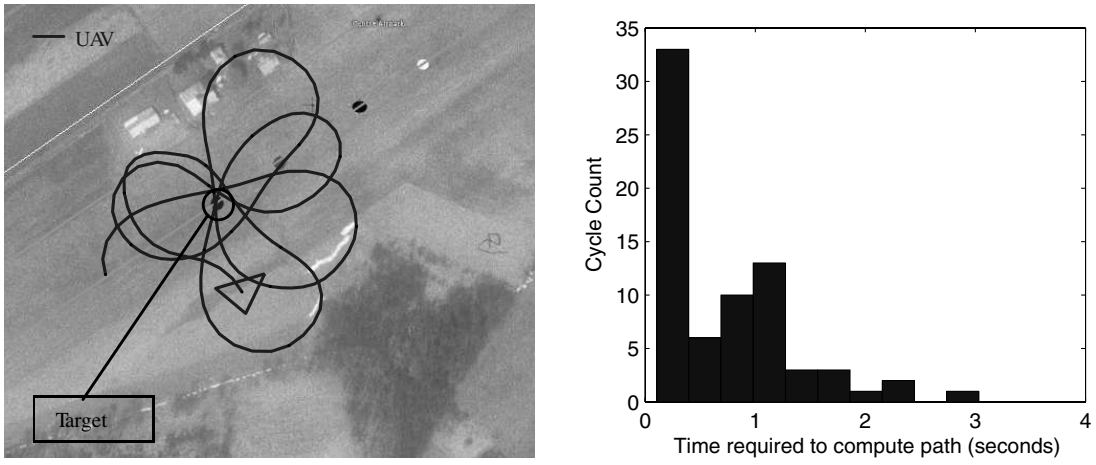
2. *Stationary Target in Steady Winds*

Figure 12a shows the results of observing a stationary target in a 5-kt wind coming from the west (the left side of the figure). The figure-8 pattern seen in simulation emerges in the flight test. Also note that all of the turns are made into the wind: this minimizes the ground-track turn radius. Comparing the path update time histogram (Fig. 12b) with the calm-wind case shows that compensating for wind requires slightly more processing time, which is to be expected. Configuration 2 was used in both of the results shown for the stationary-target tests (7 nodes, 20-s horizon time, and 4-s update interval). Configuration 1 produced similar results and was able to operate in real time. The only significant difference was the amount of processing time required. For configuration 1, the onboard camera has a view of the target 41% of the time, and for configuration 2, it achieves 40% time coverage.

3. *Tracking a Moving Ground Target*

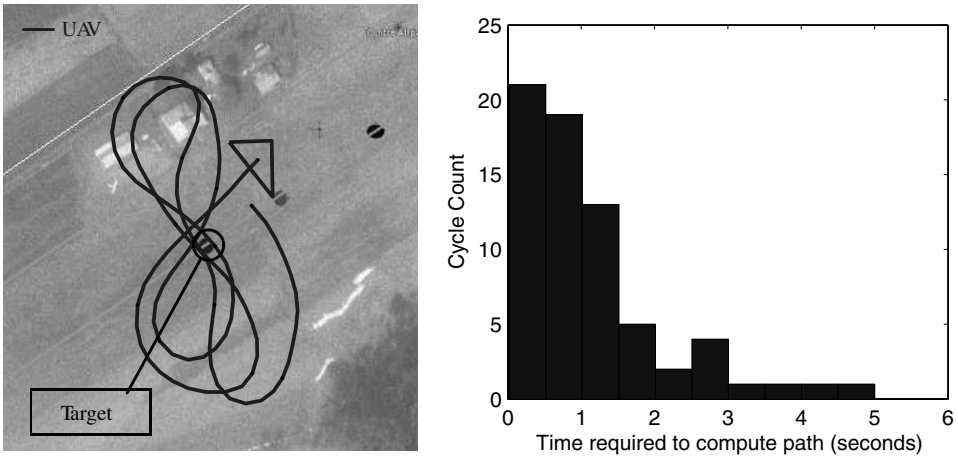
For the second set of flight tests, a spare Piccolo autopilot was used as a mobile target. The laptop connected to the ground station runs a server on the 802.11 wireless network that mirrors the data received from the autopilots. The computer onboard the tracker UAV connects to this server over the 802.11 network and is able to receive all data from the target autopilot.

Both a person walking with the target Piccolo and a truck driving down the adjacent road were used to test the mobile target-tracking ability of the path planner. We found that a person walking was not fast enough to produce much variation in the path (other than the motion of the path center) of the UAV compared with a stationary target. Figure 13a shows a typical result. The person, represented by a dashed line, is walking along the edge of the field heading toward the northeast. The UAV, shown as a solid line, repeatedly circles over the person as he moves down the field. The settings used for this test are a 30-s horizon time, 10 segments, and 6-s update interval. For this case, the real-time performance is bordering on exceeding the time requested for the update interval. When this happens, the path planner continues to operate, but the UAV traverses a longer portion of the path before it is updated. In extreme cases, the UAV may reach the end of the path before a new update is available and may turn around to the beginning. This would mean that real-time operation is not possible. Figure 13b shows that the average time to generate a



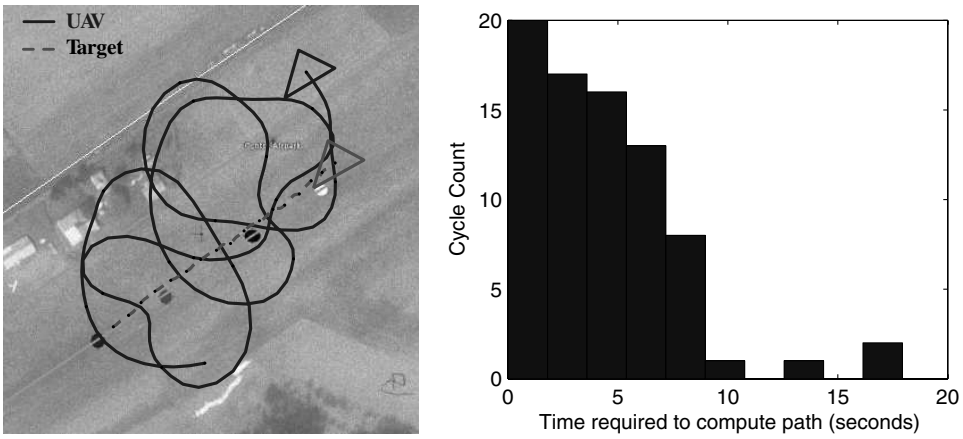
a) Ground track **b) Path-generation time**

Fig. 11 Stationary-target observation in calm winds.



a) Ground track b) Path-generation time

Fig. 12 Stationary-target observation with a 5-kt wind from the west.

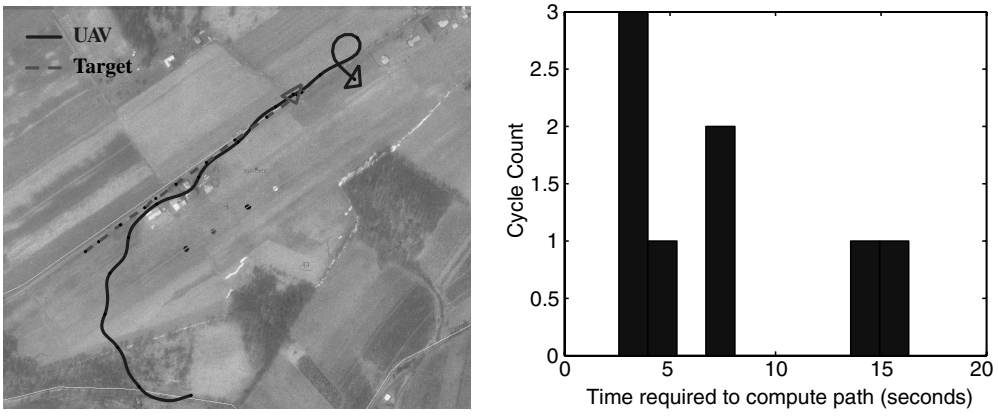


a) Ground track b) Path-generation time

Fig. 13 Observation of a walking person.

new path is approaching the update interval. This is likely due to the inclusion of target motion: the optimization requires more time to converge with more variables in play. The path planner achieved 38% time coverage for the walking person and 40% coverage with the person running.

To test the performance of the path planner with a faster moving ground target, a pickup truck driving down a road adjacent to the airfield was tracked. The target Piccolo autopilot was placed in the truck. Because of the hilly terrain, there was a limit on how far away the truck could be before the signal to the target autopilot was lost. Therefore, only a limited amount of road was available for the tracking test. The truck drove at 20 mph. Figure 14a shows the ground track of the truck as a dashed line and the UAV as a solid line. This test mainly shows the behavior of the path planner when the target is stopping and starting, because there was not enough space to safely track the truck with the UAV. There are still some real-time implementation problems, as shown by the timing plot in Fig. 14b. The cause of the long processing times was the truck stopping



a) Ground track b) Path-generation time

Fig. 14 Observation of a moving truck (20 mph); time required to generate a new path.

completely to turn around. This means that the initial guess at the correct path would be fairly bad and the optimization would take longer to converge. A shorter update interval may be advantageous when tracking a mobile target because it would allow for more timely updates when the target changes direction. As the path planner is currently configured, it assumes the target will always continue following its current heading. Using a predictive target model similar to the road-data model discussed in Sec. IV.A would bring improvement.

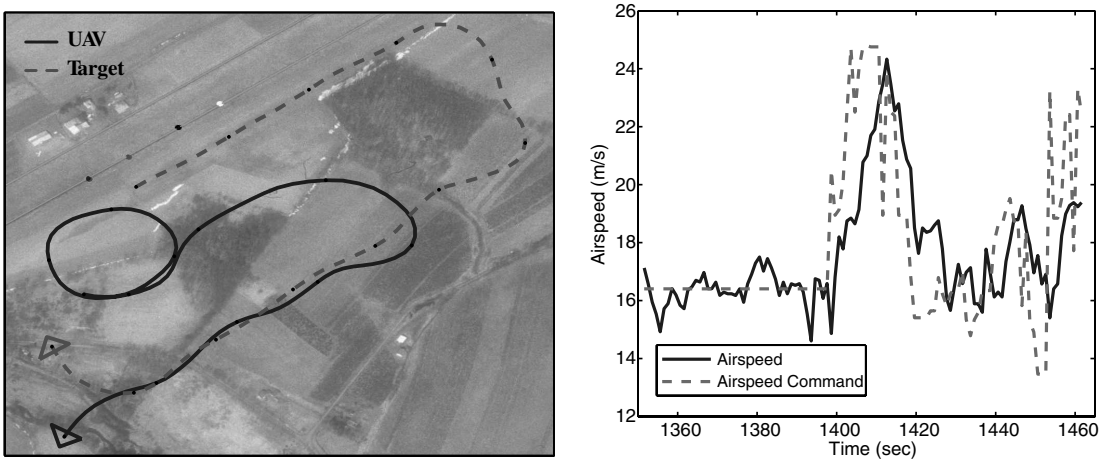
4. *Tracking a Flying UAV from Above*

Because ground space with which to test the tracking of a vehicle is limited surrounding the airfield, a ground vehicle was simulated by using a UAV flying at a lower altitude than the tracking UAV. Using the same method discussed in Sec. V.B.3 to upload the target's GPS position to the tracker UAV, the target UAV was flown on various flight plans 300 ft below the tracker UAV. This way, we had much greater freedom to test the performance of the path planner when tracking a moving target.

The results shown in this section required a change in the implementation compared with the previous results. Speed commands were not sent in the preceding stationary- and slow-moving-target cases, because the optimal observation airspeed is the minimum safe airspeed. For the following cases with a fast target, speed commands in addition to the path waypoints were sent to the autopilot. We found from an initial test that fixed speeds would not work even when the observation and target UAVs' speeds were the same. Note that speed commands cannot be updated when the path is being calculated. Because the target's speed does not vary quickly, this limitation did not cause a problem. The limitation could be overcome by using an intermediary program to send commands to the autopilot.

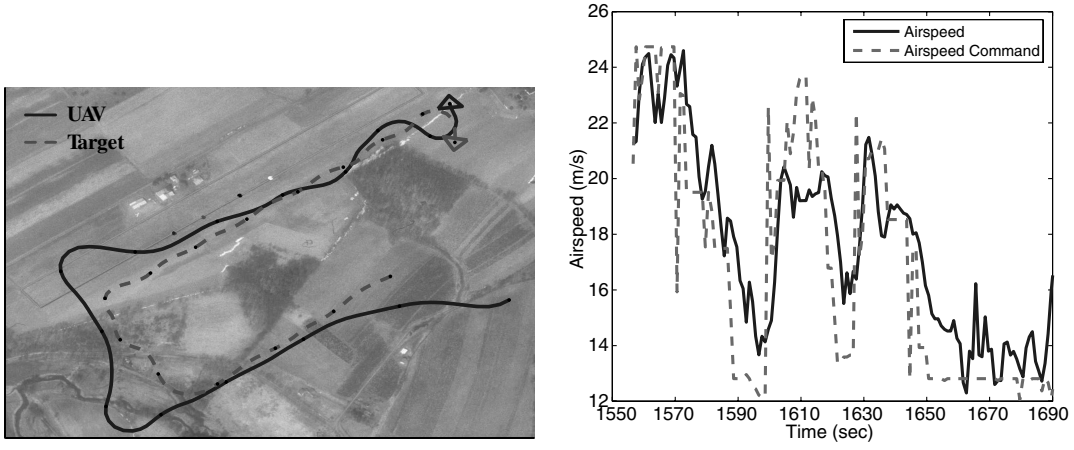
The initial test tracked a UAV around a simple rectangular pattern. For this test, the tracker UAV was allowed to match the speed of the target UAV. Figure 15 shows the start of the maneuver. The tracker UAV is shown by the solid line and starts from its parking orbit. The tracker smoothly intercepts the target UAV and assumes position above it. The path planner generates a path that commands the UAV to speed up to close the distance to the target. This can be seen at 1400 s in Fig. 15b. As the tracker UAV approaches the target, it is commanded to slow down, as seen at 1420 s. When the target UAV turns the corner of its flight plan, the path-planner commands the UAV to speed up again. When the target UAV turns the corner of its flight plan, the tracker UAV continues on for a short time until the path planner updates the path with the new heading of the target. Figure 16 shows the UAVs on their second time around the pattern. The tracker UAV overshoots the target when the target turns a corner, but accelerates to make up ground and decelerates to reacquire the target. This behavior can be clearly seen in Fig. 16b. Onboard video was not available for this test because the UAV lost engine power and crashed, which caused the video to be lost. However, based on the ground track, the target is estimated to be covered 35 to 45% of the time.

A following flight retested this scenario with a slightly different rectangular pattern. When the UAV was allowed to match the



a) Ground track b) Airspeed and command time history

Fig. 15 Initial acquisition of the target UAV from a parking orbit.



a) Ground track b) Airspeed and command time history

Fig. 16 Tracking of the target UAV around a rectangular path.

target's speed, the coverage time was measured to be 33%. When the UAV is limited to 10 kt faster than the target, the coverage time is 25%. However, it was clear from both videos that for a significant amount of time, the target was just out of view of the bottom of the frame. The optimization and equations of motion assume that the UAV does not pitch. However, in flight, the UAV is pitched up some actual amount, which would point the camera forward. We may see better results by accounting for even a constant amount of pitch angle based on the trim attitude of the UAV at some nominal airspeed. However, new analytical derivatives would need to be calculated to include pitch attitude.

The next results show flights test with the path planner using road data. In this case, the road is simply the flight plan of the target UAV. The current position along the flight plan and speed of the target UAV is sent to the tracker UAV. The path planner then uses this information in predicting the future path of the target when generating a new trajectory. A figure-8 flight plan was used for the target UAV. The first scenario has the tracker UAV limited to 15.4-m/s (30-kt) minimum airspeed, and the target can travel at 10.3 m/s (20 kt). Figure 17a shows the ground track, and Fig. 17b shows that for the majority of the time, the tracker UAV is at its minimum-allowed airspeed. For this scenario, the target coverage time was 27% of the total time.

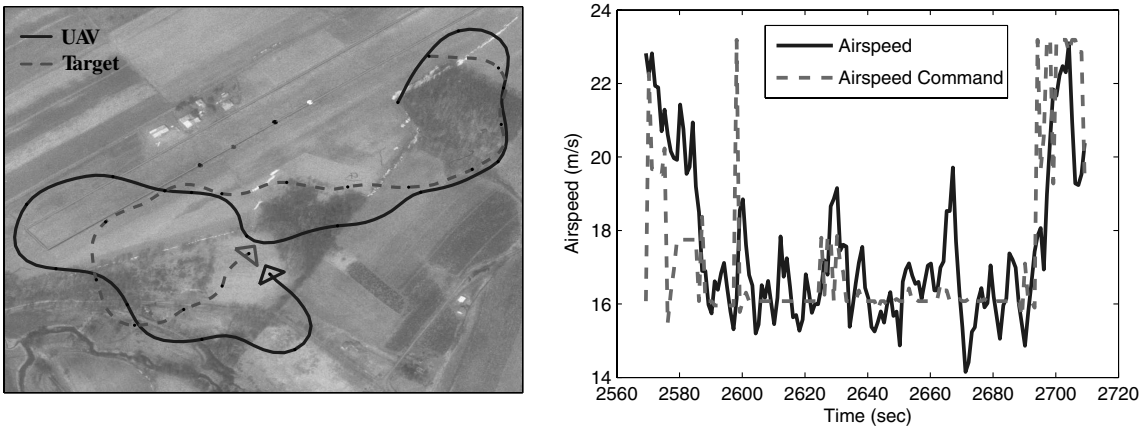
The second scenario tested with the figure-8 road was the case when both target and tracker UAV can fly at the same speed. The ground track of the UAVs is shown in Fig. 18a and is continued in Fig. 18b. The speed of the target UAV was set to 15.4 m/s (30 kt) to match the minimum-allowed airspeed of the tracker from the previous scenario. The average path-calculation time for this and the previous scenario is 1.5 s. Although the target is tracked closely, the tracker UAV is never able to maintain position directly over the target UAV. It seems to consistently lag approximately 2–3 s behind the target. This may be due to the latency involved in downlinking the target's position to the ground station and then resending it to the tracker UAV. This apparent lag could be corrected by applying a correction to the position of the target UAV on its path proportional to its speed and the estimated latency. In this scenario, the target coverage time was 69% of the total time.

The following figures show a frame capture from the onboard video. Figure 19a shows a person walking while dragging two red balls (for increased visibility in the video). Figure 19b shows a target UAV flying 300 ft below. The person and UAV are circled in white.

VI. Discussion

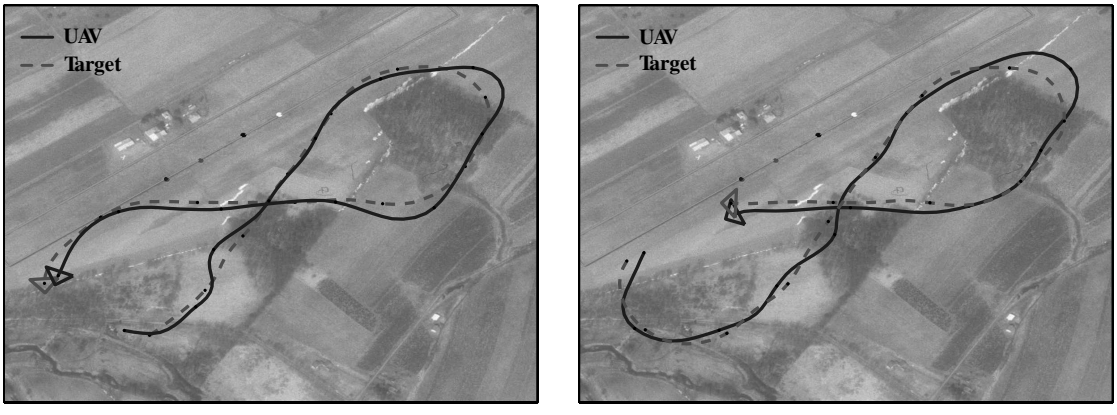
Of the two configurations flight-tested (Table 1), configuration 2 performed adequately in all cases. This configuration maintained a significant margin between path-calculation time and the update interval. The average path-calculation time for configuration 2 was around 1.5–2 s using an update interval of 4 s, which makes processor resources available for other tasks such as image processing. Configuration 1, which used more nodes and a longer horizon time, worked in the simpler stationary-target scenarios, however, its performance suffered when a mobile target was used due to its longer update interval.

To achieve the computation speed necessary to use this method in real time on this particular processor, analytical derivatives for the objective and constraint gradients are required. This hampers making even small changes to the objective or constraint functions (useful in research), because the derivatives must be recalculated, which is



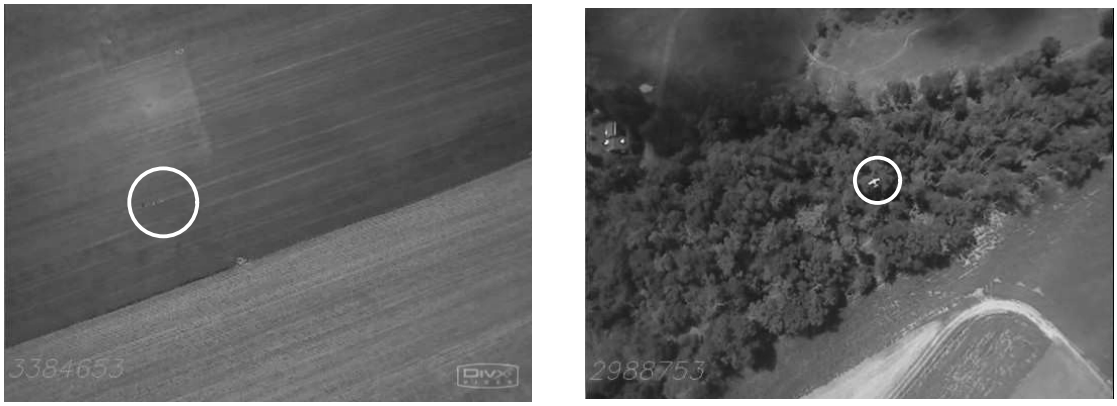
a) Ground track b) Airspeed and command time history

Fig. 17 Tracking of the target UAV around a figure-8 pattern with a 5-m/s (10-kt) speed difference.



a) Ground track b) Ground track continues

Fig. 18 Tracking of the target UAV around a figure-8 pattern in which the tracker can match the target speed.



a) Tracking a walking person b) Tracking a UAV

Fig. 19 Sample frame captures from the onboard video.

tedious and can introduce errors. However, writing analytical derivatives is not impossible, and they only need to be written once if the general mission scope of the UAV does not change.

Although the method depends entirely on the convergence of the nonlinear solver, the UAV is not physically at risk from invalid path solutions. The system is set up to send only waypoint and speed commands to the autopilot, and simple checks can be implemented to ensure that invalid commands are not used. Therefore, the UAV can be prevented from proceeding along a trajectory that would cause harm or cross boundaries.

Neglecting pitch angle does reduce observation performance. This could be corrected by including pitch in the actual observation, but a more useful fix may be to simply add pitch to the camera-orientation matrix instead of adding to the equations of motion. This would avoid adding parameters to the optimization problem, while accounting for aircraft pitch. The pitch angle could be scheduled with airspeed. If the target observation is assumed to take place at a constant airspeed, the pitch angle could be held constant over the horizon time.

Note that the algorithm is being run on a processor released in 2004. During flight tests, observed CPU usage averaged approximately 50%. Thus, with proper setup, a simple image-processing routine to track a specific target could be integrated. With the release of more powerful processors since 2004, including low-power multicore designs, it is reasonable to state that the method presented would be useful in a more complete airborne package.

VII. Conclusions

The direct collocation method is shown to be able to capture views of the targets in all flight-test scenarios and, in most cases, to maintain a reasonable amount of sensor coverage, given the limitations of the particular system tested (a single nongimbaled camera). In addition, there was great similarity between the simulated and actual flight-test results in these scenarios. This shows that despite the simplifying assumptions made, the method performs well in actual flight.

We also found that a motion model that predicts future target motion greatly enhances path-planning performance. However, the simple straight-line target-motion assumption did perform well for targets that are slow or not highly agile. Modeling wind effects is just as important as modeling target motion in any path-planning application. A moving target and a steady wind have the same effect on the UAV's path when transformed into the target's local reference frame. In addition, a steady wind can even sometimes be advantageous when observing a target, because it can be used to lessen the speed difference between the UAV and target. This advantage may only be useful for smaller low-flying UAVs such as the one used in this work.

In practice, the nonlinear solver is generally robust in terms of providing a valid solution. The number of nodes used in the optimization is critical for ensuring real-time operation of a path planner using direct transcription methods. Paths that use fewer nodes may be less optimal, but they can be updated more frequently, which is important for incorporating changing sensor data. However, a reduction in the number of nodes must be accompanied by a reduction in horizon-time length to maintain accurate dynamics interpolation. This reduction, of course, reduces target-motion anticipation. Although the method can be computationally expensive, especially for larger problem sets, we believe that we have shown a viable and useful compromise between the number of optimization parameters and the real-time observation results achieved.

Acknowledgments

This research was funded by the Pennsylvania State University's Applied Research Laboratory's (ARL) Exploratory and Foundational Program. In addition, aircraft, equipment, and facilities were provided by the ARL Intelligent Systems Laboratory.

References

[1] Rysdyk, R., "Unmanned aerial vehicle path following for target observation in wind," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 5, 2006, pp. 1092–1100. doi:10.2514/1.19101

[2] Frew, E. W., Xiao, X., Spry, S., McGee, T., Kim, Z., Tisdale, J., Sengupta, R., and Hendrick, J. K., "Flight Demonstrations of Self-Directed Collaborative Navigation of Small Unmanned Aircraft," *AIAA 3rd 'Unmanned-Unlimited' Technical Conference, Workshop, and Exhibit*, Vol. 2, AIAA, Reston, VA, Sept. 2004, pp. 1108–1121.

[3] Dobrokhodov, V. N., Kaminer, I. I., Jones, K. D., and Ghabcheloo, R., "Vision-Based Tracking and Motion Estimation for Moving Targets Using Small UAVS," *2006 American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 14–16 June 2006, pp. 1428–1433.

[4] Hargraves, C. R., and Paris, S. W., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, July–Aug. 1987, pp. 338–342. doi:10.2514/3.20223

[5] Dickmanns, E. D., and Well, K. H., "Approximate Solution of Optimal Control Problems Using Third Order Hermite Polynomial Functions," *Proceedings of the IFIP Technical Conference*, Springer–Verlag, London, 1974, pp. 158–166.

[6] Herman, A. L., and Conway, B. A., "Direct Optimization Using Collocation Based on High-Order Gauss–Lobatto Quadrature Rules," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 592–599. doi:10.2514/3.21662

[7] Coverstone-Carroll, V., and Prussing, J. E., "Optimal Cooperative Power-Limited Rendezvous with Propellant Constraints," *Journal of the Astronautical Sciences*, Vol. 43, No. 3, 1995, pp. 289–305.

[8] Tang, S., and Conway, B. A., "Optimization of Low-Thrust Interplanetary Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 3, 1995, pp. 599–604. doi:10.2514/3.21429

[9] Enright, P. J., and Conway, B. A., "Discrete Approximations to Optimal Trajectories Using Direct Transcription and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 4, 1992, pp. 994–1002. doi:10.2514/3.20934

[10] Herman, A. L., and Conway, B. A., "Optimal Spacecraft Attitude Control Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 5, 1992, pp. 1287–1289. doi:10.2514/3.20983

[11] Horie, K., and Conway, B. A., "Optimal Aeroassisted Orbital Interception," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 5, 1999, pp. 625–631.

[12] Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T., "Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 4, 2004, pp. 586–594. doi:10.2514/1.11168

[13] Betts, J. T., and Cramer, E. J., "Application of Direct Transcription to Commercial Aircraft Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 151–159. doi:10.2514/3.56670

[14] Goto, N., and Kawable, H., "Direct optimization methods applied to a nonlinear optimal control problem," *Mathematics and Computers in Simulation*, Vol. 51, No. 6, 2000, pp. 557–577. doi:10.1016/S0378-4754(99)00145-7

[15] Zhao, Y. J., "Optimal Patterns of Glider Dynamic Soaring," *Optimal Control Applications and Methods*, Vol. 25, No. 2, 2004, pp. 67–89. doi:10.1002/oca.739

[16] Zhao, Y. J., and Qi, Y. C., "Minimum Fuel Powered Dynamic Soaring of Unmanned Aerial Vehicles Utilizing Wind Gradients," *Optimal Control Applications and Methods*, Vol. 25, No. 5, 2004, pp. 211–233. doi:10.1002/oca.744

[17] Borrelli, F., Subramanian, D., Raghunathan, A. U., and Biegler, L. T., "MILP and NLP Techniques for Centralized Trajectory Planning of Multiple Unmanned Air Vehicles," *2006 American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 14–16 June 2006, pp. 1–6.

[18] Misovec, K., Inanc, T., Wohletz, J., and Murray, R. M., "Low-Observable Nonlinear Trajectory Generation for Unmanned Air Vehicles," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 3, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2003, pp. 3103–3110.

[19] Milam, M. B., Mushambi, K., and Murray, R. M., "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2000, pp. 845–851.

[20] *Open Source Computer Vision Library* [online library], http://www.intel.com/technology/computing/opencv/ [retrieved 7 July 2008].

[21] Miller, J. A., Minear, P. D., Niessner, A. F. Jr., DeLullo, A. M., Geiger, B. R., Long, L. N., and Horn, J. F., "Intelligent Unmanned Air Vehicle Flight Systems," *Journal of Aerospace Computing, Information, and Communication*, Vol. 4, No. 5, 2007, pp. 816–835.

[22] Sinsley, G. L., Miller, J. A., Long, L. N., Geiger, B. R., Niessner, A. F., and Horn, J. F., "An Intelligent Controller for Collaborative Unmanned Air Vehicles," *IEEE Symposium Series in Computational Intelligence*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Apr. 2007, pp. 139–144.

[23] Trucco, E., and Verri, A., *Introductory Techniques for 3-D Computer Vision*, 1st ed., Prentice–Hall, Upper Saddle River, NJ, 1998.

[24] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013