Cooperative Control of Small UAVs for Naval Applications

Isaac I. Kaminer*, Oleg A. Yakimenko*, Vladimir N. Dobrokhodov*, Mariano I. Lizarraga*, and Antonio M. Pascoal**

* Department of Mechanical and Astronautical Engineering, Naval Postgraduate School, Monterey CA
** Department of Electrical Engineering, Instituto Superior Técnico, Lisbon Portugal

Abstract - This paper addresses the development of a cooperative control algorithm used to launch and recover a fleet of small UAVs from a ship. The key features of the algorithm include trajectory generation for multiple UAVs that accounts for their aerodynamic characteristics and guarantees deconfliction, particularly on final approach, and path following control for multiple UAVs to track these trajectories. The proposed control approach is sufficiently flexible to allow for multiple formation configurations and sequential landing patterns. The paper includes simulation results and ends with conclusions and recommendations for future work.

I. INTRODUCTION

This paper addresses the development of a shipboard autoland system for multiple small unmanned air vehicles (UAVs). The typical mission scenario includes a ship underway that has launched and now needs to recover a team of small UAVs. (Specific UAVs considered are Silver Foxes (SFs) produced by ACR of Tucson, AZ.) It is assumed that initially the UAVs are flying in formation towards the ship. The approach for sequential autoland of the UAV formation presented in this paper includes:

i) real-time trajectory generation for each UAV so as to bring it to the top of the glideslope from its place in the formation during a specified time slot (Segment 1 on Fig.1). These trajectories must guarantee deconfliction; furthermore, the time slots are selected to provide the UAV team aboard the ship sufficient time to retrieve each UAV from the net;

ii) real-time glideslope generation to bring each UAV from the top of glideslope to the center of the net, moving with the ship (of course this Segment 2 is actually being computed first to provide the final point for the Segment 1);

iii) a control strategy to force each UAV to track the trajectories developed in steps 1 and 2.

This paper is organized as follows. Section 2 discusses the implementation of the direct method of optimal control developed in [1] and modified here to guarantee sequential collision-free arrival of multiple UAVs in step 1. Section 3 addresses the construction of a stabilized glideslope that brings each UAV to the center of the net. Sections 4 addresses control system design and simulation results. Section 5 discusses the hardware implementation and presents hardware-in-the-loop simulation results. The paper ends with the conclusions and a description of future work.

II. NEAR-OPTIMAL REAL-TIME TRAJECTORIES GENERATION

This section presents the theory and algorithms for real-time trajectory generation. This theory is first presented for the case of a single UAV as in [1], after which it is shown how the trajectory optimization problem can be reformulated for a group of UAVs flying in formation to provide sequential collision-free landing (glideslope capturing).
A. Optimal Problem Formulation

In very general terms, the problem of optimal control that we consider in this paper is that of determining the optimal UAV trajectory from an initial to a final given point in space, satisfying UAV dynamics, and restrictions imposed on the state variables, controls and their derivatives [1]. “Optimal” means that it minimizes either the integral function on the state variables, controls and their derivatives [1].

The trajectory parameters are determined numerically at N points equally spaced over the virtual arc with increments \( \Delta \tau = \tau_f (N - 1)^{-1} \). This corresponds to the time intervals

\[
\Delta t_j = \frac{\tau_f}{\sum_{j=1}^{N} \sqrt{(x_{i,j+1} - x_{i,j})^2}}, \quad (j = 1, N - 1).
\]

With the values \( \Delta \tau \) and \( \Delta t_j \), the parameter \( \lambda \) (4) is calculated at each step. The explicit laws for UAV’s coordinates (2), with the velocity (3) calculated at the corresponding time instants,
uniquely determines all the motion parameters: γ(t), μ(t), φ(t) and n(t) [1].

F. Parameter Optimization Problem

Thus, the optimization algorithm may be presented as follows. Using an arbitrary value of the virtual arc length τ and a set of free polynomials’ coefficients K (when n > n’), we calculate the reference polynomial (2). Then, also using arbitrary initial guess of optimization parameter, defining throttle history – arc T τ ∗, we integrate equation (3) over the interval 0 ≤ τ ≤ f with the integration step Δτ.

Then, with the help of corresponding relations, we obtain the values of state coordinates and controls [1]. At the end of the trajectory, we compute the functional J and the penalty function G compounded of weighted discrepancy in the final velocity and constraints violations.

As a result, we obtained a minimization problem in the following form

\[ \hat{\Xi}_{\text{opt}} = \arg \min_{\Xi \in \Xi} J(\hat{\Xi}) , \]

where \( \hat{\Xi} = \{ \tau_j; K; \tau_f \} \), and ε is the predefined tolerance.

This problem can be effectively solved with the help of any zero-order method, e.g. the Nelder-Mead downhill simplex algorithm or the Hooke-Jeeves pattern direct search algorithm. Of course, both algorithms should be modified in order to search the functional extremum only when penalty function is less than specified value (correspondent scripts were written in MATLAB). The weights of the penalty function G are chosen heuristically to ensure the specified accuracy when matching the terminal value of the UAV’s velocity and all the restrictions.

Basically, minimizing arc \( \tau_j \) corresponds to the time-optimal control, but any other parameter, different from the time \( \tau_j \), or even the superposition of several parameters can be used as a functional. This is one of the main advantages of the direct method. Two other advantages of the proposed near-optimal trajectories generation routine are: i) intuitively understandable analytic presentation of the near-optimal solution, and ii) convergence robustness.

G. Extension to Multiple UAV Maneuvering

In practice it is worth keeping the dimension of the optimization problem as low as possible. For a single UAV we managed to efficiently obtain a solution with only three variable parameters (K in this case is represented by a single parameter related to the initial planar curvature of the trajectory).

In case of multiple UAVs, say M UAVs, the dimension of the problem increases to 3M. As to additional constraints, two sets of constraints should be added (correspondent discrepancies added to the penalty function G).

The first one introduces a scheduled (with interval Δ) arrival at the top of the glide slope:

\[ t_f^j - t_f^j = t_f^j - t_f^{j-1} = ... = t_f^{M-1} - t_f^M = \Delta . \]

The second one takes care of the collision avoidance (assuming minimal separation of E):

\[ \min_{j,k=1,...,M} \sum_{i=1}^M (x_i^j - x_i^k)^2 \geq E^2 . \]

H. Numerical Examples

As an example, Fig.2 illustrates a certain flexibility of the reference polynomials to compute a coordinated attack by several UAVs. Figure 3 shows that this approach can be easily extended to compute the entire landing approach trajectory (skipping the glideslope segment and bringing the UAV right before the net). Finally, Fig.4 shows an example of joint optimization for scheduled arrival at the top of the glideslope for a cluster of three UAVs originally flying in the compact formation.

III. GLIDESLOPE TRAJECTORY GENERATION

The autoland concept used in this paper calls for each UAV to capture and track a stabilized glideslope that ends in a net placed across the ship’s helicopter landing pad (recall Fig.1). (Stabilized glideslope represents a straight line that is attached to the center of the net and follows the ship’s translational motion while maintaining fixed spatial orientation.) The net is equipped with two high precision
DGPS receivers that provide the UAV with information on ship’s motion. While DGPS is capable of providing lateral (x-y) position information to within 1m, the achievable accuracy in the vertical direction is only 3m. Therefore, in addition the net contains a barometric altimeter that measures the net’s altitude with respect to sea level. This measurement is used to compute each UAV’s relative barometric altitude by comparing it with the onboard barometric measurement. This approach leads to vertical errors on the order of 1m, i.e. sufficient for the autoland task at hand. Construction of the glideslope with respect to a net deployed on the ship is discussed next.

Let the vector \( \hat{z}_n = [0, 0, 1]^T \) be a unit vector that points in the local up direction and let \( p_1 \) and \( p_2 \) denote positions of two points in the net expressed in \( \{U\} \) as shown in Fig.5. Then the unit vector that points along the main diagonal of the net is given by \( \hat{d}_n = (p_2 - p_1)\|p_2 - p_1\|^{-1} \). Using these two coplanar vectors (\( \hat{z}_n \) and \( \hat{d}_n \)), a vector normal to the net that points in the direction opposite the ship’s motion can be obtained as follows: \( \hat{n}_n = \frac{\hat{z}_n \times \hat{d}_n}{\|\hat{z}_n \times \hat{d}_n\|} \). The position of a middle point in the net is \( p_c = 0.5(p_2 - p_1) \).

Let \( \gamma' \) determine the angle between the glideslope and the horizon (as shown in Fig.6), and let \( d \) specify the desired length of the glideslope.

Then the top of the glideslope is given by

\[
p_t = p_c + dR_y\hat{n}_n,
\]

where \( R_y \) represents an elementary rotation matrix about the local \( y \) axis by an angle \( \gamma' \). Now the glideslope trajectory has the form

\[
p_t(s) = p_o + T(s - d), \tag{5}
\]

where \( s \) denotes the physical (not virtual) path length along the glideslope, and \( T = -\hat{n}_n \). Note, that \( p_t(0) = p_o \), \( p_t(d) = p_c \), and \( s = v_t \), where \( v_t \) is the magnitude of the desired velocity along the glideslope.

### IV. CONTROL SYSTEM DESIGN AND TESTING

The development of a control system - to track desired paths - followed the methodology for the design of path following controllers presented in [2], [3] which can be briefly explained as follows. The trimming trajectories of the UAV are easily seen to be helices, which can degenerate into straight lines and circumferences. Furthermore, the trimming trajectories can be parameterized in terms of the UAV’s total speed, flight path angle, and yaw rate. It is also possible to show that tracking of a trimming trajectory or path (in case one is not interested in meeting tight time constraints) is equivalent to driving a conveniently defined error vector to zero. In the case of path following, the error vector captures: i) the “lateral deviation” of the vehicle to the path to be followed, projected along a Serret-Frenet frame with origin at the closest point on the trajectory, ii) the mismatch between the desired orientation of the UAV at trimming and the actual orientation, and the iii) the mismatch between desired and actual linear and rotational velocities. By noticing that the linearization of the generalized error dynamics about any trimming trajectory is time invariant, it is then straightforward to see that the problem of control system design for accurate path following can be cast in the framework of gain scheduled control theory. In this context, the vehicle’s linear speed, yaw rate, and flight path angle play the role of scheduling variables that interpolate the parameters of linear controllers designed for a finite number of representative trimming paths. This circle of ideas can then be extended to deal with general paths that are not necessarily trimming paths, by resorting to the formalism of linear parametrically varying systems. (The details are not shown here.)

Let \( p_{UAV} \) denote current position of the UAV expressed in \( \{U\} \). As discussed in [2] the first step is to define the Serret-Frenet frame \( \{F\} \) along the glideslope trajectory. The unit vector \( T \) defined in the previous section is used to define the \( x \)-axis in \( \{F\} \). Then the \( y \)-axis is defined by a unit vector \( N = \left([-T_x, T_z, 0]^T, \|[-T_x, T_z, 0]\|^1 \right)^{-1} \) and \( B = T\times N \). Now the rotation matrix from \( \{U\} \) to \( \{F\} \) is

\[
R = [T, N, B]^T.
\]

Next, one must obtain the nearest point \( p_t(s) \) on the glideslope trajectory. Therefore, it is necessary to find the parameter \( s \), that minimizes the norm of the position error

\[
s = \arg \min_s \|p_c(s) - p_{UAV}\|. \tag{6}
\]

Simple algebra shows that

\[
s = 0.5\left(T^T(p_{UAV} - p_c) + (p_{UAV} - p_c)T\right) + d.
\]

Then from (5) \( p_c(s) = p_c + T(s - d) \) and the position error vector expressed in \( \{U\} \) is

\[
p_c(s) = (p_c - p_{UAV}) + T(s, -d).
\]
Now the position error resolved in \( \{ F \} \) is given by

\[
P_{\text{e}}(s) = \frac{r}{u} R p_{\text{e}}(s),
\]

which eventually simplifies to

\[
P_{\text{e}}(s) = \left[ 0, N^T(p - p_{\text{UAV}}), B^T(p - p_{\text{UAV}}) \right]^T.
\]

Similarly it can be shown that the velocity error vector \( v_{\text{e}}(s) = \frac{r}{u} R \left( T^v v_{\text{e}} - v_{\text{UAV}} \right) \) resolved in \( \{ F \} \) is given by

\[
v_{\text{e}}(s) = \left[ v_{\text{e}} - T^v v_{\text{UAV}}, -N^T v_{\text{UAV}} - B^T v_{\text{UAV}} \right]^T,
\]

where \( v_{\text{UAV}} \) represents the UAV’s velocity vector in \( \{ U \} \).

Figure 7 shows the resulting control system architecture used by each UAV to track a given trajectory. This architecture was imposed by the hardware implementation where the autopilot resides onboard of each UAV while the path following controllers are executed on the ground station (see Section 6). As shown in Fig.7, simple PID controllers were used for tracking. Each uses position and velocity errors in the lateral and vertical channels and was sufficient to provide good tracking performance and stability robustness.

Figure 8 presents results of a nonlinear simulation of an autoland by a single UAV in calm seas (three plots shown are the UAV trajectory, Euler angles history and controls time history). Autoland initiation point was 700m away and 100m up from the initial net location (while UAV approaches it, the net moves together with the ship at about 700m away from its original location). In this case the net impact error was 0.72m. On the other hand, as can be seen in Fig.9 in the presence of realistic winds, sensor noise, and of a sea state 3 (waves up to 2m in height, ship’s heave and roll up to 5 deg) the final error for the same initial conditions increases to 6.3m. Further analysis showed that the inaccuracies inherent in the onboard DGPS are responsible for 70% of this error. Therefore, the replacement of a new sensor suite should greatly improve net impact errors.

Finally, Fig.10 shows the results of preliminary Monte-Carlo simulation for a single UAV with an arbitrarily picked initial point (with respect to the top of the glide-slope). The impact points for these runs are also presented. As seen almost all of them occur within the net span.

V. HARDWARE-IN-THE-LOOP SIMULATION

The autoland system was tested in the hardware-in-the-loop simulation provided by Cloud Cap. This simulation employs a stand alone computer that simulates the UAV motion and interfaces directly with Piccolo autopilot. Thus the autopilot flies the simulated model while its interface with the ground station via a wireless link remains unchanged.

Figure 11 shows the hardware architecture of the shipboard autoland system. Each of the Silver Fox UAVs is equipped with Piccolo autopilot produced by Cloud Cap Inc. The Naval Postgraduate School autoland system interfaces with Piccolo ground station which can control up to five UAVs simultaneously. (For the description of the complete system see [4]).
Figure 12 includes results of an autoland in the hardware-in-the-loop simulation in the presence of wind, sensor noise and sea state 3. These closely matched the results obtained in the nonlinear simulation above.

V. CONCLUSION

This paper reported a development of an integrated guidance and control system for sequential autoland of a UAV formation. In particular, building on the previous work of the authors, direct methods were shown to provide a useful methodology for generation of sequential landing trajectories; while path following algorithms were shown to do a reasonable job tracking these trajectories. It should be noted that results reported in this paper represent work in progress. Namely, a complete autoland system for a single UAV has been developed and extensively tested in simulation. Flight tests are currently under way. Furthermore, a trajectory generation algorithm that brings each UAV from its place in formation to the top of the glideslope at a given time slot has also been developed and tested in simulation. Current work addresses the development of the complete system that includes both trajectory generation and tracking from initial position in formation all the way to net impact.

REFERENCES