2002-6

Autonomous Hierarchical Control of Multiple Unmanned Combat Air Vehicles (UCAVs)

Timothy McLain
Mechanical Engineering Department, Brigham Young University, mclain@byu.edu

Randal W. Beard
Department of Electrical and Computer Engineering, Brigham Young University, beard@ee.byu.edu

See next page for additional authors

Follow this and additional works at: https://scholarsarchive.byu.edu/facpub

Part of the Mechanical Engineering Commons

Original Publication Citation

BYU ScholarsArchive Citation
McLain, Timothy; Beard, Randal W.; Li, Sai-Ming; Boskovic, Jovan D.; Seereeram, Sanjeev; Prasanth, Ravi; Amin, Jayesh; and Mehra, Raman K., "Autonomous Hierarchical Control of Multiple Unmanned Combat Air Vehicles (UCAVs)" (2002). All Faculty Publications. 1919.
https://scholarsarchive.byu.edu/facpub/1919

This Conference Paper is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Autonomous Hierarchical Control of Multiple Unmanned Combat Air Vehicles (UCAVs)*

Sai-Ming Li, Jovan D. Boskovic, Sanjeev Seereeram, Ravi Prasanth, Jayesh Amin, Raman K. Mehra
Scientific Systems Company
500 W. Cummings Park, Ste 3000
Woburn, MA
eliot@ssci.com

Randal Beard
444 Clyde Building
Brigham Young Univ.
Provo, Utah 84602
beard@ee.byu.edu

September 18, 2001

Abstract

In this paper we present a hierarchical control scheme that enables multiple UCAVs to achieve demanding missions in hostile environments autonomously. The objective is to use a swarm of UCAVs for a SEAD type mission: fly the UCAVs in a formation to an enemy territory populated with different kinds of threats, collect enemy information or destroy certain targets, and return to the base, all without human intervention. The scheme is an integration of four distinct components, including: (i) High level Voronoi diagram based path planner to avoid static threats; (ii) Low level path planner to avoid popup threats; (iii) Differential flatness based trajectory generator to generate dynamically feasible trajectory; (iv) Semi-globally stable formation control algorithm to maintain the formation. The scheme was implemented in Matlab and demonstrated very effective path planning, trajectory generation, and formation flying capabilities. We also developed an interface from Matlab to IWARS, a high fidelity battlefield simulation environment developed by Boeing. This enabled us to study the effectiveness of our scheme under various battle scenarios using IWARS.

For American Control Conference 2002

*This work is supported by the ONR through contract N0014-99-C-0324.
1 Introduction

In recent years, there is a growing interest in employing Unmanned Combat Air Vehicles (UCAVs) for various military missions. UCAVs are particularly suited for tasks that are considered too demanding or dangerous for pilots. One typical example is the Suppression of Enemy Air Defense (SEAD) mission, in which the objective is to fly to the enemy territory to destroy, or suppress, their surface-based air defenses, such as radars or surface-to-air missile (SAM) sites. The mission is critically important to any air operation as only after this can a more comprehensive air attack be initiated. To increase the chance of mission success, typically multiple UCAVs are flown together so that in case one is lost during the mission, the remaining ones can still complete the mission. Moreover, multiple UCAVs offer extra capabilities over single UCAVs such as cooperative geolocation of targets. Apart from the obvious advantage of no risk to human pilots, UCAVs also allow more aggressive maneuver, significant weight savings, lower costs, potential for superior cooperation, and opportunity for new operational paradigms. However, to realize these advantages, the UCAVs must possess high level of autonomy so that they can operate without human intervention.

A typical SEAD mission is depicted in Figure 1. By stationary or static threat we mean threats that can be detected before the mission, and their positions are available to the control scheme at any time. On the other hand, during the mission, the UCAVs may encounter popup threats that cannot be detected before the mission starts, such as radars that are turned on periodically and can be detected only when that is the case. Such mission is highly dangerous as the target to be suppressed are usually heavily protected by anti-aircraft weapons, if it is not one itself. To achieve the mission with minimal human intervention, the UCAVs need to possess highly autonomous capabilities at different levels, including path planning, trajectory generation, and formation control. Previous work typically focus at only one of these capabilities, and hence do not offer a complete solution to the problem. In this paper we present an hierarchical control scheme that integrates all the required components to solve the UCAV control problem in a modular fashion.

An architecture of our scheme is shown in Figure 2. The first layer is the High-Level Path Planner, which takes static threat and target information from the command center. This planner plans a path that represents a compromise between the cost involved in exposing to the threats (which is related to the distance of the path from the threats), and the cost involved in fuel expense (which is related to path length). It determines a path from a graph search through the edges of a Voronoi diagram formed from the threat locations. The vertices of the diagram become the waypoints passing to the next layer. The second layer, Low-Level Path Planner, is used to plan a finer path from the current position to the waypoint received from the High-Level path planner. This layer is constantly checking if there is a popup threat along the current path leading to the next waypoint. If there is, this layer will plan a path that avoids the popup threat, while still reaching the next waypoint. A conformal mapping approach is used to generate the path: First the popup threat is bounded by an ellipse. Then it is mapped to a new space by conformal mapping where the region outside the threat becomes a convex set in this space. A threat-avoiding path can then be obtained in this space easily. Sub-waypoints are
generated along this path by sampling, and passed to the trajectory generation layer. Due to dynamic and input constraints, the UCAVs can only follow a given state trajectory when such constraints are satisfied. The role of the trajectory generation layer is to generate a feasible trajectory for the UCAVs to follow, and compute the corresponding control input for the UCAV leader. The position and input of the UCAV leader are passed to the formation controller of all the UCAV followers, where they will compute the appropriate control input for each follower such that the formation configuration is maintained despite the maneuvers made by the leader.

In section 2 we will first review some related work in the area of autonomous control of multiple UCAVs to put our effort in context. Then we will describe each of the layers in our control scheme in detail, along with simulation example whenever it is appropriate. Section 7 describes our work on implementing an interface between Matlab Simulink and IWARS, an interactive warfare simulation environment developed by Boeing. This allows us to study the effectiveness of our control scheme in a high fidelity battle simulation environment. The paper is ended by several concluding remarks in Section 8.

2 Related Work

Path Planning has been the focus of research in the robotics community [1, 2]. However, the research in path planning for UCAVs is relatively limited. An overview of the path planning problem, issues, and solutions is given by Goldman in [3]. Pellazar [4] proposed the use of genetic algorithms for air vehicle route planning under constraints. A Newton mechanics-inspired
Figure 2: Hierarchical controller architecture for multiple UCAVs.

approach was proposed by Bortoff [5] that models the UCAV trajectory by a series of point masses connected by springs. The target exerts an attractive force on the masses while the threats exert repulsive forces on them. The final path will depend on the relative strength of these two forces assumed in the algorithm. The High Level path planning algorithm used in this paper will be based on the Voronoi diagram-based approach developed by McLain and Beard [6] due to its generality and efficiency.

Trajectory Generation, just like path planning, has attracted a lot of attention in the robotics community in the past several decades. The main application area is in the control of robot manipulator. Typically, the geometries of the robot and environment are used to find a set of feasible joint coordinates between the initial point and the goal. Then a smooth joint trajectory is formed using conservative estimates of joint accelerations[7]. In aircraft control, an optimal control approach is more common [8]. However, computing optimal control, especially under
constraints, tend to be very computationally intensive and is not suited for real-time control. Recently, geometric approach that exploits certain dynamical properties of the system concerned, has been proposed [9, 10]. They are found to be applicable to a large class of UCAV dynamics, and will be used in the trajectory generation layer described in this paper.

In [11] the basic formation flight control problem was introduced. Since then, several authors have considered the kinematic and aerodynamic coupling effects introduced by close formation flying [12, 13, 14, 15]. In [11, 12, 13], linear formation control design method was proposed that solves the problem around a fixed operating point. However, the linear model used in the control design may not be valid for large initial conditions, which can result in unacceptable performance. In [16, 17] the authors used the model of the relative distances dynamics to design a feedback-linearizing control law. However, the model contains a singular point and the implementation of the feedback-linearizing controller may result in ill-defined control inputs. In [18] we proposed a globally stable formation control algorithm in two dimensions based on a new control error formulation in the inertial coordinates. It was shown that the error dynamics has a singularity in the desired position space, rather than in the state space. Hence the control algorithm is always valid as long as the desired positions are chosen properly. We will use this algorithm in the formation control layer in our scheme.

3 High Level Path Planner

In this section we describe the High Level Path Planner in detail. The planner is responsible for generating a path that avoids the static threats, and reaches all the targets in the prescribed order. We adopt the approach proposed by McLain and Beard [6], and the readers are referred to the reference for more details. Under this approach, a Voronoi diagram is first constructed based on the locations of the static threats. Figure 3 shows the Voronoi diagram for a given set of threats and targets. In the figure, the circles and squares represent the static threats and targets respectively, with the exception that the zeroth square is the initial location of the UCAV leader. The targets are numbered according to the order in which they are to be visited. Notice that the Voronoi diagram is created without regard to the initial formation location or the target location.

To generate an initial path, the initial position of the lead UCAV is connected to the three closest node of the Voronoi diagram. Hence we have a graph that connects the lead UCAV to the target through the edges of the Voronoi diagram. The problem here is to choose an “optimal” path in this graph. For that a cost is assigned to each edge of the Voronoi diagram, and the total cost of a particular path is the sum of the cost of all the edges in that path. The cost associated with traveling along a particular edge $i$ consists of two components: the cost $J_{t,i}$ due to the risk incurred, which is related to the proximity of the edge to the threats; and the cost $J_{f,i}$ that relates to the fuel required, which depends on the length of the edge. It is assumed the threats are mainly of radar type and the strength of a UCAV’s radar signature is proportional to $1/d^4$, where $d$ is the distance to the threat. Instead of integrating this cost along each edge, which is a computational intensive procedure, it is approximated by an weighted average of the
cost at three points on the edge.

When calculating the total cost $J_i$ for edge $i$, the designer can adjust a parameter $k$ that represents a balance between the risk cost $J_{ki}$ and the fuel cost $J_{fi}$ to vary the aggressiveness of the planner:

$$J_i = kJ_{ki} + (1 - k)J_{fi}, \quad 0 \leq k \leq 1.$$  

After the cost associated with each edge is fixed, we can determine the optimal path by searching for the lowest cost path. The Dijkstra’s algorithm [19] is used for this purpose here. The algorithm require the graph to be a directed one, i.e., a direction has to be specified for each edge of the graph. In this implementation we assign the direction of all the edges to be toward the target. Figure 4 shows a path generated by this algorithm for the threat/target configuration shown in Figure 3. The nodes of this path is passed as waypoints to the Low Level Path Planner in the hierarchical control scheme in Figure 2.
4 Low Level Path Planner

In this layer the objective is to determine a path that connects the current position of the lead UCAV to the next waypoint, while avoiding any popup threat should they appear. Here we adopt a simple approach that works in the two dimensional case, and is illustrated in Figure 5. The top left corner of Figure 5 shows the inertial frame, the current position, the next waypoint, and the popup threat. If there is no popup threat, or when the popup threat does not cross the line-of-sight path, this planner will return the line-of-sight path that connects the current position and the next waypoint. In the case where the line-of-sight path is not feasible, first a coordinate change is applied such that the threat region becomes a unit circle centered at the origin (z-domain in Figure 5). For example, suppose that the threat region is defined by the ellipse:

\[ \{ x : (x - x_c)^T R^{-1} (x - x_c) \leq 1 \}, \]

where \( R \) is a positive definite matrix and \( x_c \) is the center. The coordinate change is given by:

\[ \hat{x} = R^{-1/2} (x - x_c) \]
and, in the new coordinates (i.e. $z$-domain), the start state $x_s$ and the desired final states $x_f$ are:

$$\hat{x}_s = R^{-1/2} (x_s - x_c), \quad \hat{x}_f = R^{-1/2} (x_f - x_c).$$

We now view the $z$-domain as the complex plane and apply a conformal transformation:

$$\omega = \frac{1}{z}$$

to interchange the threat region’s interior and exterior ($\omega$-domain in Figure 5). The problem now becomes one of finding a path between the current spatial state and the desired final spatial state that lies inside the unit circle. Since the conformal transformation takes $\infty$ to 0, trajectories that come close to zero in the $\omega$-domain will be mapped back into unbounded trajectories. Therefore, a short path in the original space will correspond to one that is close to the unit circle in the $\omega$-domain. Hence we choose a path that is close to the circle (see Figure 5). This path is sampled to generate sub-waypoints that connects the current position and the next waypoint. Finally, the sub-waypoints are mapped back to the initial coordinates and pass to the next layer for trajectory generation.

If there are multiple threats as shown in the bottom left corner of Figure 5, then an ellipsoid of minimum volume (area) that contains all the threats is computed first to reduce the situation to the one just discussed. Computing a minimum volume ellipsoid containing $m$ given ellipsoids is a linear matrix inequality problem and can be solved efficiently.

It is also possible to view threats as regions of varying risk rather than as regions to be avoided completely. To formulate trajectory generation problem in this case, let us attach to each trajectory $x(t)$, $t \in [0, T]$ a risk value:

$$J(x) = \sum_{i=1}^{M} \int_{0}^{T} r_i(x(t))dt$$

where $r_i$ is a positive function describing the risk associated with the $i$th threat and we assume that the contributions are additive. Typically, the functions $r_i$ roll-off away from the threat region. The trajectory generation problem is to plot a course from the current state to the final state that minimizes the risk functional $J$ subject to system dynamics and constraints. This is a problem in variational calculus and there are many solution procedures [8].

Figure 6 shows the final path followed by the UCAV formation under the threat/target configuration shown in Figure 3, but with two popup threats represented by the red filled circles in the figure. For the sake of clarity, only the lead UCAV is shown. Notice that the path followed by the UCAVs in Figure 6 is slightly different from the initial path shown in Figure 4. The final path is deviated from the initial path to avoid the two popup threats. This path replanning is conducted online as the popup threats are detected only when the UCAVs were about to cross them.
5 Trajectory Generation Layer

Due to dynamical constraints and physical limits on control authority, not all trajectories connecting two sub-waypoints can be followed by the UCAV. It is the responsibility of the Trajectory Generation Layer to generate a feasible state trajectory, as well as a set of inputs for the UCAV leader. Recently, there is a lot of interest in the trajectory generation problem for a special class of systems called differentially flat systems. The important property of differentially flat systems is that we can express all states and inputs in terms of those outputs and their finite derivatives. More precisely, a nonlinear system

\[ \dot{x} = f(x, u) \]

is differentially flat if we can find flat outputs \( z \) of the form

\[ z = \zeta(x, u, \dot{u}, \ldots, u^{(L)}) \]

such that

\[ x = x(z, \dot{z}, \ldots, z^{(L)}) \]
\[ u = u(z, \dot{z}, \ldots, z^{(L)}), \]
Figure 6: Final path from the Low-Level Path Planner.

Differentially flat systems are useful in situations where explicit trajectory generation is required. Since the behavior of flat systems is determined by the flat outputs, we can plan trajectories in the flat output space, then map these to appropriate inputs. A common example is the kinematic car, where the $x, y$ position of the rear wheels is the flat outputs of the system [20, 21]. This implies that all feasible trajectories of the car can be determined by specifying the trajectory of the rear wheels only.

The UCAV/autopilot models considered under this study are assumed to be first order for heading and Mach hold, and second order for altitude hold. Let $(X_i, Y_i, \psi_i, V_i, h_i)$ represent the inertial position, heading angle, velocity, and altitude of the $i$th aircraft, then the configuration variable for the $i$th aircraft is given by $\mathbf{x}_i = (X_i, Y_i, \psi_i, V_i, h_i, \dot{h}_i)^T$. The dynamics of the $i$th aircraft is given by

\begin{align}
\dot{X}_i &= V_i \cos \psi_i \\
\dot{Y}_i &= V_i \sin \psi_i \\
\dot{\psi}_i &= \frac{1}{\tau_\psi} (\psi_{\alpha_i} - \psi_i)
\end{align}

(1) (2) (3)
\[
\dot{V}_i = \frac{1}{\tau_V} (V_{ci} - V_i) \\
\dot{h}_i = - \left( \frac{1}{\tau_{ha}} + \frac{1}{\tau_{hb}} \right) \dot{h}_i + \frac{1}{\tau_{ha} \tau_{hb}} (h_{ci} - h_i),
\]  

where \( \psi_{ci}, V_{ci}, \) and \( h_{ci} \) are the heading, velocity, and altitude commands issued to the autopilots, and \( \tau_{ai} \) are constants and are assumed to be the same for all aircraft in the formation. Denote the state of the UCAV leader by \( \mathbf{x}_F = (X_F, Y_F, \psi_F, V_F, h_F, \dot{h}_F)^T \). Then its dynamics is given by (1)-(5) with the subscript \( i \) replaced by \( F \).

For the aircraft model considered here, a flat output can be chosen as \( z = (z_1, z_2, z_3)^T = (X_F, Y_F, h_F)^T \). The initial condition and the first derivative of \( z \) is available from the measurement. The final condition of \( z \) is just the location of next sub-waypoint. The first derivative of the final condition of \( z \) is obtained by assuming that the UCAV will reach a predefined nominal speed at the next sub-waypoint. Let \( t_o \) be the current time. The time span \([t_o, t_F]\) for the trajectory from the current location to the next sub-waypoint is specified by computing the time \( t_n \) required to travel between these two points in a straight line with the nominal speed, and \( t_F = t_o + t_n \). Let \( z : [t_o, t_F] \to \mathbb{R}^3 \) be a parametrized path from the current configuration to the desired configuration at the next waypoint. To avoid an infinite dimensional problem, let each element of \( z \) be given by a third order polynomial in \( t \in [t_o, t_f] \), i.e.,

\[
z(t) = \left( \sum_{i=0}^{3} c_{i+1} t^i, \sum_{i=0}^{3} c_{i+5} t^i, \sum_{i=0}^{3} c_{i+9} t^i \right).
\]

Then the coefficients \( c_i \) are solved using the boundary conditions of \( z \) at \( t_o, t_f \). The full state trajectory and nominal input for the UCAV leader can then be lifted from \( z \).

6 Formation Control Layer

In this section we describe the Formation Control Layer of our hierarchical scheme. The details of the formation control algorithm can be found in [18]. Here we will just state the problem in analytic terms, and present the formation control algorithm and the theoretic result obtained in [18].

Relative Position Dynamics

Figure 7 shows a rotating reference frame affixed to the follower’s instantaneous position and aligned with the follower’s velocity vector. From Figure 7, the inertial position of vehicle 1 (the leader) can be expressed in terms of the inertial position of vehicle 2 (the follower) and its relative position from the body frame of vehicle 2 as

\[
X_1 = X_2 + x \cos \psi_2 - y \sin \psi_2 \\
Y_1 = Y_2 + x \sin \psi_2 + y \cos \psi_2
\]  

(6)

(7)
Then the relative coordinates $x$ and $y$ can be expressed in terms of the inertial positions of the two vehicles and the heading angle of vehicle 2 as

$$ x = (X_1 - X_2) \cos \psi_2 + (Y_1 - Y_2) \sin \psi_2 $$

$$ y = -(X_1 - X_2) \sin \psi_2 + (Y_1 - Y_2) \cos \psi_2 $$

Differentiating with respect to time and substituting for the turn rate $\dot{\psi}_2$ from (3), we have

$$ \dot{x} = -\frac{y}{\tau \psi_2} \dot{\psi}_2 - V_2 + V_1 \cos(\psi_1 - \psi_2) + \frac{y}{\tau \psi_2} \dot{\psi}_{2c} $$

$$ \dot{y} = \frac{x}{\tau \psi_2} \dot{\psi}_2 + V_1 \sin \psi_e - \frac{x}{\tau \psi_2} \dot{\psi}_{2c} $$

The formation control problem for the above model is stated below.

**Formation Control Problem:** Design a Formation-Hold Autopilot (FHA) that generates $V_{2c}$ and $\psi_{2c}$ for vehicle 2 such that the relative position errors converge to zero asymptotically, even in the face of maneuvers by the lead vehicle (vehicle 1).

The formation control problem for formations with more than one follower can be stated in the same way for each follower, though the problem of how to avoid collision during formation initialization have to be addressed separately. Previous work on this problem rely on linearization of the formation error dynamics and the control law has a singularity in the error space [16, 17]. This is not desirable as the implementation of such control law may result in ill-defined control input. The main idea behind the formation controller in [18] is to derive an alternative form of relative position error using vehicle positions in the inertial frame. The dynamic of this new error has a special structure that allows us to design a singularity-free stabilizing controller.
We will show that stabilizing this error dynamics is equivalent to tracking the desired relative position between the two vehicles.

**A New Error Definition**

From (6), (7), we have

\[
\begin{bmatrix}
X_1 - X_2 \\
Y_1 - Y_2
\end{bmatrix}
= \begin{bmatrix}
\cos \psi_2 & -\sin \psi_2 \\
\sin \psi_2 & \cos \psi_2
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
= R(\psi_2) \begin{bmatrix}
x \\
y
\end{bmatrix},
\]  
(12)

where

\[
R(\theta) \triangleq \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]  
denotes a rotation matrix from the body frame \((x, y)\) of vehicle 2 to the inertial frame \((X, Y)\) (Figure 7). Define an error \(E\) as

\[
E = \begin{bmatrix}
E_1 \\
E_2
\end{bmatrix}
\triangleq \begin{bmatrix}
X_1 - X_2 \\
Y_1 - Y_2
\end{bmatrix}
- R(\psi_2) \begin{bmatrix}
x_r \\
y_r
\end{bmatrix}.
\]  
(13)

From (12) and (13), we have that

\[
E = R(\psi_2) \begin{bmatrix}
x - x_r \\
y - y_r
\end{bmatrix}.
\]  
(14)

Hence, if \(E = 0\), both \(x(t) - x_r\) and \(y(t) - y_r\) will tend to zero as well, as \(R(\psi_2)\) is non-singular for all \(\psi_2\), Therefore, to achieve the desired relative position between the two vehicles, we can simply design a control law that stabilizes \(E(t)\).

**Error Dynamics**

Upon differentiating (13) twice with respect to time, we have

\[
\ddot{E} = \begin{bmatrix}
\eta_1 \\
\eta_2
\end{bmatrix} + G \begin{bmatrix}
V_{2c} \\
\dot{\psi}_{2c}
\end{bmatrix},
\]  
(15)

where

\[
\eta_1 \triangleq \ddot{X}_1 + \frac{1}{\tau V_2} V_2 \cos \psi_2 + (V_2 \sin \psi_2 - \frac{1}{\tau \psi_2} (x_r \sin \psi_2 + y_r \cos \psi_2) + (x_r \cos \psi_2 - y_r \sin \psi_2) \dot{\psi}_2),
\]  
(16)

\[
\eta_2 \triangleq \ddot{Y}_1 + \frac{1}{\tau V_2} V_2 \sin \psi_2 + (-V_2 \cos \psi_2 + \frac{1}{\tau \psi_2} (x_r \cos \psi_2 - y_r \sin \psi_2) + (x_r \sin \psi_2 + y_r \cos \psi_2) \dot{\psi}_2),
\]  
(17)

and

\[
G \triangleq \begin{bmatrix}
\frac{1}{\tau V_2} \cos \psi_2 & \frac{1}{\tau \psi_2} (x_r \sin \psi_2 + y_r \cos \psi_2) \\
\frac{1}{\tau V_2} \sin \psi_2 & \frac{1}{\tau \psi_2} (-x_r \cos \psi_2 + y_r \sin \psi_2)
\end{bmatrix}.
\]  
(18)

**Control Law**

The control variables \(V_{2c}\) and \(\dot{\psi}_{2c}\) in (15) are chosen as

\[
\begin{bmatrix}
V_{2c} \\
\dot{\psi}_{2c}
\end{bmatrix} = G^{-1} \{ - \begin{bmatrix}
\eta_1 \\
\eta_2
\end{bmatrix} - k_{E1} \dot{E} - k_{E2} E \},
\]  
(19)
where $k_{E1} = \text{diag}\{k_{E11}, k_{E12}\}, k_{E2} = \text{diag}\{k_{E21}, k_{E22}\}$, and $k_{Eij} > 0$ for $i, j = 1, 2$. Then the closed-loop dynamic for $E$ is
\[
\dot{E} + k_{E1}\dot{E} + k_{E2}E = 0,
\]
which is asymptotically stable. Hence $\lim_{t\to\infty} E(t) = 0$, which implies $x(t) - x_r$ and $y(t) - y_r$ will converge to zero asymptotically.

It should be noticed that the determinant of $G$ is given by
\[
|G| = -\frac{x_r}{\pi \psi_2 \psi_2}. \tag{21}
\]
Hence for $x_r \neq 0$, $|G| \neq 0$ and $G$ is non-singular. Therefore, unlike the FHA presented in the previous section, this autopilot does not have any singularity in the state space. Although this FHA has a singularity at $x_r = 0$, this can be avoided by choosing $x_r$ appropriately.

To implement this control law, the heading command $\psi_2$, is obtained by integrating $\dot{\psi}_2$. The initial value $\psi_2(t_0)$ can be set to $\psi_2(t_0)$ to avoid large transients.

The properties of the above control law are summarized in the following theorem:

**Theorem 1** [18] The control law (19) is globally asymptotically stable for (15), and results in
\[
\lim_{t\to\infty}(x(t) - x_r) = \lim_{t\to\infty}(y(t) - y_r) = 0.
\]

Extensive simulation study was conducted to demonstrate the effectiveness of the control law (19), and was reported in [18]. We have also extended the algorithm to the three-dimensional case and the result can be found in [22].

### 7 IWARS Simulink Interface

The Interactive Warfare Simulation (IWARS) system is a many-on-many desktop simulation tool developed by Boeing. The tool is designed to facilitate analyses and trade-off studies for weapon-system development. The IWARS simulation framework consists of objects, scenarios and map overlays. Objects may include aircraft, ships, radar sites, SAM sites, missiles, etc., all of which can be defined by the associated dynamics, sensors, and weapons. IWARS allows the analysts to define custom scenarios, i.e., how many objects of each type, and the initial position, speed, heading, and flight plan of each object. IWARS also allows freedom to modify and build map overlays, thus permits the user to simulate a battle at different geographical locations. IWARS allows analysts to construct simulated weapons systems and to define system capabilities and behaviors for large numbers of mixed assets. Users can control the assets, such as aircraft or missiles, interactively or by defining simple reactive strategies. However, for more advanced control strategies such as the hierarchical scheme described in this paper, IWARS is not flexible enough to implement such complex strategies. Moreover, since Simulink is the most versatile simulation environment for advance control design and development, it is beneficial to develop the control scheme in Simulink, and interface the output of the Simulink model with
IWARS to study their effectiveness in more realistic battle scenario. We have developed such
an interface between Simulink and IWARS, and is described below.

Using IWARS requires running two software programs: Global Object Simulation (GOS)
and Control Panel. The GOS models a battle environment by simulating and updating the
state of all objects in a scenario. The environment may include aircraft, ships, missiles and
other user-definable objects as well as their associated sensor suites and weapon loadouts. The
Control Panel displays the view of the battle area and provides the user interface to control the
game. Since the GOS maintains the state of all the objects in the simulation, it is sufficient to
develop an interface between GOS and Simulink.

The Interface design is based on: (i) UDP using the Boeing Unified Message Passing (UMP)
system for individual UCAV control and, (ii) TCP using the Boeing Packet Communications sys-
tem for the Mission Planner and the Formation Controller interface. The simulink S-Functions
were coded in C++ based on this interface. Two Simulink S-Function blocks were developed:
Mission Planner (MP) and Formation Controller (FC). The functionalities of these two blocks
are described below in detail.

The Mission Planner (MP) implements a two way communication with the IWARS Global
Object Simulation (GOS). It receives the target tracks (threats) and the simulation state infor-
mation from the GOS, and sends waypoint and formation commands to the GOS (to relay to
the FC). The Formation Controller (FC) implements code to receive formation and waypoint
commands from the GOS (sent by the MP). It requests and takes control of individual UCAVs
for formation flying, and instantiated blocks of FC get UCAV state updates from the MAT-
LAB/Simulink environment and relay these to the GOS for IWARS Control Panel updates. FC
can release control of a UCAV to the GOS for other controllers (including human) if not involved
in formation flying. The MP and FC blocks can be dragged and dropped into user Simulink
models and can reside on two different simulations running on separate dedicated machines.

8 Conclusion

In this paper we presented a hierachical control scheme that enables multiple UCAVs to achieve
demanding missions in hostile environments autonomously. The scheme is an integration of four
distinct components, including: (i) High level Voronoi diagram based path planner to avoid
static threats; (ii) Low level path planner to avoid popup threats; (iii) Differential flatness
based trajectory generator to generate dynamically feasible trajectory; (iv) Semi-globally stable
formation control algorithm to maintain the formation. The scheme was implemented in Matlab
Simulink and demonstrated very effective path planning, trajectory generation, and formation
flying capabilities. We also developed an interface from Matlab Simulink to IWARS, a high
fidelity battlefield simulation environment developed by Boeing. This enabled us to study the
effectiveness of our scheme under various battle scenarios using IWARS.
Acknowledgement

We would like to thank Andy Miller and his colleagues at Boeing for technical assistance in developing the IWARS Simulink interface, and Tim McLain at Brigham Young University for providing us with the Voronoi diagram-based path planner.

References


