

# An Optimal Timing Approach to Controlling Multiple UAVs

X.C. Ding, M. Powers, M. Egerstedt, and R. Young

**Abstract**—In this paper we address the problem of having a single operator control a team of unmanned aerial vehicles (UAVs). This is achieved by having the team execute a leader-follower coordinated behavior, where the leader is responsible for the execution of the high-level mission. The operator interacts with the system by selecting a leader and a decision support mechanism is provided whereby the system computes the best choice of leader in the current situation. This feedback is obtained through a novel, receding horizon optimal timing control that computes an on-line estimate as to the relative merits of selecting different vehicles as leaders. The method is implemented in a dynamic, 3D simulation environment, illustrating the soundness of the proposed approach.

## I. INTRODUCTION

One of the biggest challenges facing the successful deployment of unmanned aerial vehicles (UAVs) in unstructured environments is the level of human involvement needed to carry out the mission. In fact, control and coordination of UAVs typically involve a *many-to-one* mode of operation in that multiple operators are needed to control a single UAV. The explicit purpose of this work is to invert this relationship, i.e. to enable a single pilot to control and coordinate multiple unmanned vehicles. This will enable pilots to operate UAV teams much more effectively.

In this paper we envision a scenario in which the human operator (the pilot) is flying along-side a team of UAVs that are to execute a given task. The pilot must control the team of UAVs while maintaining control of his own plane, which implies that only a few modes of interaction with the UAV team are feasible in order not to overload the pilot with data and with decision tasks. We approach this problem by essentially allowing the pilot to interact with the UAV team along two basic modes of operation, namely (1) *autonomous leader-follower mode*, and (2) *pilot-controlled leader-follower mode*. In the first of these two modes, one UAV is designated as the leader (by the pilot) and its job is to ensure a proper execution of the overall mission in an autonomous fashion. The remaining UAVs will position themselves with respect to the leader UAV in order to maintain a proper inter-agent separation. The pilot can, while in the autonomous leader-follower mode, switch between leader UAVs as well as transition to the other mode

of operation in which the pilot is directly controlling the leader UAV. An example of what the coordinated behaviors will look like is given in Figure 1, in which a three-vehicle scenario is simulated in the Player/Gazebo 3D simulation environment used to evaluate the proposed control and coordination methodology.

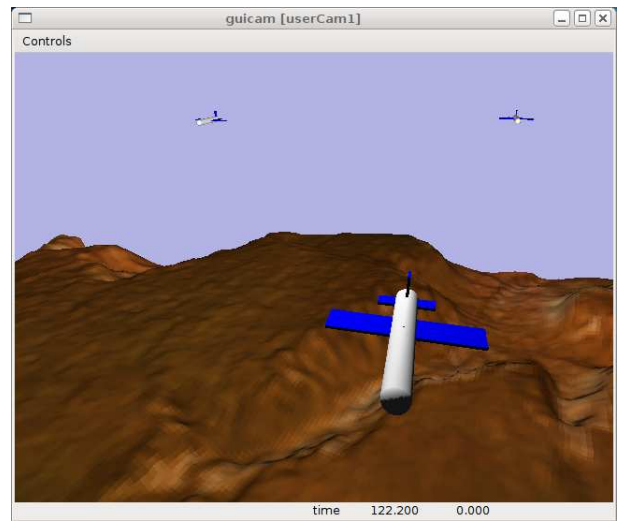


Fig. 1. UAVs in Player/Gazebo 3D Simulation Environment.

The problem of controlling a collection of mobile agents is not new, and a number of approaches have appeared during the last decade. For instance, in [1],[2],[3] and [4], a framework was established in which a tele-operator was incorporated into the control system as a so-called motor schema affecting the vehicles. In particular, the human operator was incorporated into the control system in two different ways. The first way was one in which the tele-operator can obtain total control of a single robot. The other way was to use the human operator as a supervisor that modifies the gains and parameters of existing control laws without taking complete control of the vehicle. It should be noted that under this approach, the operator produced the same schema for all the robots in the network. A similar idea was pursued in [7], where the effect of human supervisory control on swarming networks was explored, where *supervisory control* refers to the process of intermittently interacting with a system in order to select modes of operation. Other work in which humans were interacting with teams of mobile robots included [11], in which an human-robot interaction interface was produced for a large network of autonomous mobile robots. In particular, a centralized user interface was developed that collects data and sends command to

Xu Chu Ding and Magnus Egerstedt are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: {ding,magnus}@ece.gatech.edu

Matt Powers is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: mpowers@coc.gatech.edu

Ryan Young is with the Advanced Technology Center, Rockwell Collins, Cedar Rapids, IA 52498, USA. Email: syoung@rockwellcollins.com

the network. The internal state, local neighbor positioning, and global robot positioning were displayed in the proposed interface. The user was able to send a way-point through the interface to a single robot to produce a velocity vector that moved the robot to the way-point.

What is novel with the approach proposed is not that a human operator is controlling an individual vehicle in a semi-autonomous multi-vehicle setting, but the fact that actable information is transmitted back to the operator in term of the relative merit associated with different leader selections. This merit is computed based on a fast optimal timing control algorithm in which the cost associated with switching between leaders can be established.

## II. PROBLEM DESCRIPTION

The fundamental building block behind the proposed approach concerns what formations to use in a given situation. For example, in [8], it is pointed out that the robots should be spread out when navigating and exploring free-space, while a more tight formation is preferred when negotiating cluttered environments. In [5] this type of observation is made concrete by proposing a hybrid control architecture in which the robot team switches between different formations as a reaction to environmental changes.

In this paper we will not focus on formations but rather on leaders, i.e. the switching strategy will be based on selecting leader UAVs to be under direct pilot control. Furthermore, in the proposed pilot decision support framework, optimal leader selection as well as the leader switching strategy is provided as *decision aid* to the pilot. However, leader selection will not be forced in the sense that they will not occur autonomously. Instead the pilot will make the decision as to what (if any) leader to control directly.

In the remainder of this section, we define the problem that must be solved in order to successfully provide the pilot with actable information in terms of what leader is most beneficial in a given situation.

### Switch-Time Optimization Problem

It is necessary to introduce some concepts and algorithms involving the switch-time optimization problem (see for example [10]). They are useful for developing optimal-control based ranking algorithm that automatically quantifies the impact of changing leaders.

Consider an autonomous hybrid dynamical system in the form of:

$$\dot{x} = f_{\alpha_i}(x), t \in [\tau_{i-1}, \tau_i], i = 1, 2, \dots, N+1, \quad (1)$$

with a given time horizon  $\tau_{N+1} = T$  and initial condition  $x_0 = x(0) = x(\tau_0)$ . The functions  $f_{\alpha_i} : \mathbb{R}^n \rightarrow \mathbb{R}^n$  are continuous differentiable and referred to as modal functions. In our case, each modal function corresponds to a particular choice of leader in the UAV network.  $\tau_i$  is the time instant when the mode transition is triggered and system switches from mode  $i$  to mode  $i+1$ . The sequence of mode  $\{\alpha_1, \alpha_2, \dots, \alpha_{N+1}\}$  is referred to as the mode sequence and is denoted by  $\bar{\alpha}$ . Given the mode sequence of the system, we call the vector

$\bar{\tau} = [\tau_1, \tau_2, \dots, \tau_N]^T$  the switching time vector. A valid switching time vector must follow the piecewise linear inequality constraint:  $0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq \tau_{N+1} = T$ . The set of all switching time vectors that satisfy this constraint is denoted by  $\Lambda$ . Hence  $\Lambda := \{\bar{\tau} = [\tau_1, \dots, \tau_N]^T : 0 = \tau_0 \leq \tau_1 \leq \dots \leq \tau_N \leq \tau_{N+1} = T\}$ .

Now that we have a way of characterizing how the system undergoes transitions between different modes of operation, we need to establish rules for selecting the different modes of operation. And, for that we need a performance measure that determines how well the task is being carried out. As such, let  $L : \mathbb{R}^n \rightarrow \mathbb{R}$  be a continuously differentiable function, the cost function is defined by the following equation.

$$J = \int_0^T L(x)dt. \quad (2)$$

The switch-time optimization problem is to optimize the cost functional  $J$  over the switching time vector  $\{\bar{\alpha}, \bar{\tau}\}$ .

Numerous algorithms have been developed to address the switch-time optimization problem. An example can be found in [10]. Unfortunately, these algorithms are not feasible for the pilot decision support framework under consideration here. The main reason for this is that the time horizon over which a mission is specified may be way too long to make algorithms computationally feasible. In particular, since the pilot is to be provided with information in real-time, a much more focused and cheap computational algorithm must be developed. This algorithm will be presented in the next section.

## III. REAL-TIME SWITCH-TIME OPTIMIZATION

In this section, we propose to only solve the optimization problem over a smaller look ahead window, i.e. solve the switch-time optimization problem over a receding horizon. We first present our algorithms based on a general formulation and then provide the algorithms needed to address the leader-selection problem in real-time.

### A. A Receding Horizon Approach

With the receding horizon approach, the cost function that we wish to optimize over is no longer time-independent. Instead, we use a cost-to-go function that is defined by a sliding window. Let  $T_s$  denotes the length of the sliding window, the cost-to-go function will be defined as :

$$J(t, x(t), \bar{\tau}) = \int_t^{t+T_s} L(\tilde{x}(s))ds. \quad (3)$$

The instantaneous cost  $L$  is evaluated over the future (predicated) state trajectory, denoted by  $\tilde{x}(s)$ . The future state trajectory starts at initial condition  $\tilde{x}(t) = x(t)$  for the time horizon  $s \in [t, t+T_s]$ , and it evolves according to (1) with the switching time vector  $\bar{\tau}$ .

The current and future trajectory is illustrated in Figure 2, where the solid curve represents the past trajectory and the dotted curve represents the future (or projected) trajectory. In this paper we denote the problem of optimizing the cost-to-go function at time  $t$  by  $\Pi_t$ . The approach of this paper allows the means to compute the trajectory of the switching time

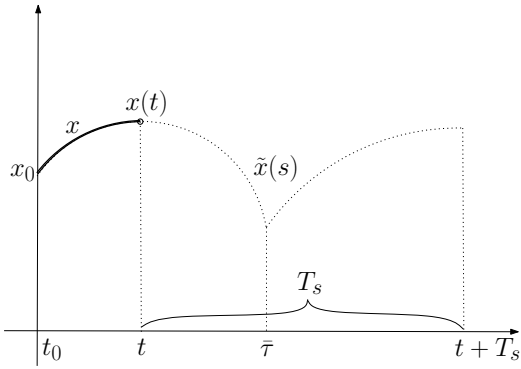


Fig. 2. Current state and the state simulated into the future.

vector, such that it is an approximation of a local optimal solution for the cost-to-go function at each time instant  $t$ . This local optimal switching time vector is a function of time  $t$  as well, and is defined as:

$$\bar{\tau}(t) = \min_{\bar{\tau}} J(t, x(t), \bar{\tau}), t \geq t_0, \quad (4)$$

for an initial time  $t_0$ . To compute trajectory  $\bar{\tau}(t)$ , we develop an iterative process so that it computes the solution for  $\Pi_{t+dt}$  assuming that the solution for  $\Pi_t$  is available. In other words, given  $\bar{\tau}(t)$ , we wish to obtain  $\bar{\tau}(t)$ .

The real-time algorithm is divided into two phases. The first phase of the algorithm is to re-compute the optimal switching times so that  $\bar{\tau}(t_0)$  is a local minimum for the problem  $\Pi_{t_0}$ . This computation must be performed on-line, whenever the system trajectory is modified by the pilot input, at which point the system clock is reset to  $t = 0$ . The time horizon for the algorithm for the first phase is set to  $[0, T]$ , where  $T$  is selected so that it is large enough to enable the convergence of the on-line algorithm. Once it converges, we set  $t_0$  to be the current time, as well as picking a sliding window length as  $T_s = T - t_0$ , and the algorithm for the second phase starts. The second phase updates  $\bar{\tau}(t)$  so that it becomes an estimate of local minimum for the next time-step.

The algorithm for the first phase is the on-line algorithm proposed in [12]. We re-iterate the algorithm here without the derivation and convergence analysis.

**Algorithm 1:** at each time  $t$ , **do**

*Step 1.* Compute  $\tilde{h}(t)$ , defined as the projection of  $-\left(\frac{\partial^2 J}{\partial \bar{\tau}^2}(t, \tilde{x}(t), \bar{\tau}(t))\right)^{-1} \frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \tilde{x}(t + \Delta t), \bar{\tau}(t))$  onto the feasible set  $\Lambda$ .

*Step 2.* Compute  $\gamma(t) := \max\{c \leq 1 \mid \bar{\tau}(t) + c\tilde{h}(t) \text{ is feasible}\}$ .

*Step 3.* Define  $h(t) := \gamma(t)\tilde{h}(t)$ , and set  $\bar{\tau}(t + \Delta t) = \bar{\tau}(t) + h(t)$ .

The gradient  $\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \tilde{x}(t + \Delta t), \bar{\tau}(t))$  is computed as follows. First the future predicated state trajectory  $\tilde{x}(s)$  is computed over the time interval  $s \in [t + \Delta t, T]$  with the switching time vector  $\bar{\tau}(t)$ . Then a projected costate trajec-

tory is computed backward:

$$\dot{\lambda}(s) = -\lambda(s) \frac{\partial f_{i+1}}{\partial \tilde{x}} - \frac{\partial L}{\partial \tilde{x}}, s \in [\tau_i, \tau_{i+1}], i = N, \dots, 1, \quad (5)$$

with the initial condition  $\lambda(\tau_{N+1}) = 0$ . The gradient is then determined by the following equation:

$$\frac{\partial J}{\partial \tau_i} = \lambda(\tau_i) \left( f_i(\tilde{x}(\tau_i)) - f_{i+1}(\tilde{x}(\tau_i)) \right). \quad (6)$$

Note that only one forward-backward computation is needed at each time instant. The above algorithm is executed at each time-step until the update direction  $h(t)$  is below a predefined bound, at which point we obtain a local minimum for  $\Pi_{t_0}$ . This concludes the first phase of the iterative process.

Then the goal becomes providing an iterative process to update the switching times so that they remain locally optimal to the sliding window at each time instant  $t$ . In the continuous-time perspective, the goal is to establish the time derivative of the optimal switching time vector  $\bar{\tau}(t)$  at time  $t$ , namely  $\dot{\bar{\tau}}(t)$ , so that optimality is conserved with respect to the sliding window. In other words, we aim to compute  $\dot{\bar{\tau}}(t)$ , so that if  $\bar{\tau}(t)$  is a solution point for  $\Pi_t$ , then  $\bar{\tau}(t + dt)$  computed by this continuous process

$$\bar{\tau}(t + dt) = \bar{\tau}(t) + \dot{\bar{\tau}}(t)dt \quad (7)$$

is a solution point for  $\Pi_{t+dt}$ , and  $\frac{\partial J}{\partial \bar{\tau}}(t + dt, x(t + dt), \bar{\tau}(t + dt)) = 0$ .

Note that if we obtain the optimal switching time vector  $\bar{\tau}(t)$  and substitute it in (3), then  $J(t, x(t), \bar{\tau}(t))$  is locally optimal and

$$\frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) = 0, \quad (8)$$

Since we wish to preserve optimality, it follows that the optimal switching time trajectory  $\bar{\tau}(t)$  also satisfies:

$$\frac{d}{dt} \left( \frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) \right) = 0. \quad (9)$$

Using the fact (9), we present the following proposition:

**Proposition 1**

Given  $\bar{\tau}(t)$  as a local minimum for  $\Pi_t$  and the second derivative of  $J$  with respect to  $\bar{\tau}$  is strictly positive. Then the time derivative of  $\bar{\tau}(t)$  can be determined by:

$$\dot{\bar{\tau}}(t) = - \left( \frac{\partial^2 J}{\partial \bar{\tau}^2}(t, x(t), \bar{\tau}(t)) \right)^{-1} M(t, \bar{\tau}(t), x(t)), \quad (10)$$

where  $M(t, \bar{\tau}(t), x(t))$  is given by

$$M(t, \bar{\tau}(t), x(t)) = \lim_{dt \rightarrow 0} \frac{1}{dt} \frac{\partial J}{\partial \bar{\tau}}(t + dt, x(t + dt), \bar{\tau}(t)).$$

The proof of this proposition uses the same reasoning as in [9], hence the detail is omitted. Throughout this paper we will make the explicit assumption that  $\bar{\tau}(t)$  is a local minimum to  $\Pi_t$  and hence that the Hessian is positive definite, which in turn implies that (10) is well-defined. This assumption may not always hold since extrema are known to not always be continuous across system parameters.

By relaxing the infinitesimal time interval  $dt$  to be a finite interval  $\Delta t$ , we obtain a discrete algorithm that is an estimate of the continuous process (7). This algorithm can be described as follows:

**Algorithm 2:** at each time  $t$ , **do**

*Step 1.* Compute  $\tilde{h}(t)$ , defined as the projection of  $-\left(\frac{\partial^2 J}{\partial \bar{\tau}^2}(t, \tilde{x}(t), \bar{\tau}(t))\right)^{-1} \frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \tilde{x}(t + \Delta t), \bar{\tau}(t))$  onto the feasible set  $\Lambda$ .

Note that the gradient formula is the same as (6), however the state and costate trajectories are computed over the time interval  $[t + \Delta t, t + \Delta t + T_s]$ .

*Step 2.* Compute  $\gamma(t) := \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma \tilde{h}(t) \text{ is feasible}\}$ .

*Step 3.* Define  $h(t) := \gamma(t)\tilde{h}(t)$ , and set  $\bar{\tau}(t + \Delta t) = \bar{\tau}(t) + h(t)$ .

The above algorithm is executed at each time instant, and it is constructed so that the optimality of the switching times is preserved. The difference between Algorithm 1 and Algorithm 2 is the time horizon used to compute the gradients and the Hessians ( $[t + \Delta t, T]$  versus  $[t + \Delta t, t + \Delta t + T_s]$ ). After an update, if a switching time is reduced to the current time  $t$ , a switching is suggested from the optimal control module to the pilot. If the pilot takes the suggestion and switches to the corresponding mode, then the first switching time is discarded from the switching time vector. If all switchings have taken place and the switching time vector becomes empty, the above algorithms can be restarted using the operation of mode insertion described in [10].

### B. Applications to a Three UAV Network

In this work, we consider a network of 3 UAVs flying in a constant altitude, connected by communication links to each other and to a remote human pilot using remote piloting software. At any point in time, the pilot may select one vehicle to act as a “leader”, navigating autonomously by a predefined mission objective. The remaining UAVs act as “followers”, falling into formation behind the leader. Thus, in this UAV network, it is possible for the pilot to switch between 3 distinct subsystems, each defined by designating a different agent as the leader. Additionally, the pilot may, at times, choose to remotely control the actions of the leader. For instance, the pilot may want to further explore an area of interest discovered during a routine mission, but not on the predefined route.

Depending on the state of the UAV network and the mission objective, it may be advantageous for the pilot to choose one UAV as the leader over another. To this end, optimal switching times can be computed on-line using Algorithm 1 and 2, and fed-back to the pilot. This information is provided to the pilot as a decision aid so that the pilot is advised to switch leader vehicles at the appropriate time.

As stated above, the majority of a mission will be completed under autonomous control by a predefined mission objective. To demonstrate how our algorithm computes the optimal trajectory for the UAV network, and as an example

mission objective, consider a cost function that evaluates progression towards completion of a surveillance mission. One strategy for surveillance of an area is to plan a path at configuration time through the area that maximizes coverage, then drive the UAV network through the path. Using this strategy, we construct a cost function that minimizes average distance from the path to the network.

The problem can be formulated as follows: Given a  $C^1$  curve  $p(t)$ , the cost function to be minimized is defined as the average distance from the network to the curve. In this case, the centroid of the network can be used. The UAV network can be modelled as a set of holonomic robotic agents, where the state of the network is defined as  $x = [x_1^T, x_2^T, x_3^T]^T$  and  $x_i \in \mathbb{R}^2$  is the position of  $i$ th robot in the network. Next, define a matrix  $C$  such that  $Cx(t)$  computes the centroid of the network. The cost-to-go function can then be defined as:

$$J(t, x(t), \bar{\tau}) = \int_t^{t+T_s} \frac{1}{2} \|C\bar{x}(s) - p(s)\|^2 ds. \quad (11)$$

It should be remarked that the computational complexity of the algorithm proposed in this paper scales linearly with the number of UAVs in the team. However, the amount of communication required may scale quadratically with the number of UAVs.

## IV. IMPLEMENTATION AND RESULTS

To demonstrate the practicality and effectiveness of the algorithms presented in this document, the pilot decision aid application described above was implemented in simulation. Care was taken to design a plausible demonstration, including implementing a distributed multi-agent system architecture, and developing within a third-party physical simulation environment. Three independent UAV agents were implemented, plus a remote pilot interface through which the calculated pilot feedback would be provided.

### A. Implementation

Each robot in the simulation was implemented as an independent agent, acting either as a leader robot or a follower robot, according to input from the pilot. At any given time, only one robot may act as a leader. The leader’s task is to follow the a priori plan, or, if commanded to do so by the human pilot, follow the pilot’s steering commands. The follower robots act to maintain an appropriate distance from the leader (a de-facto following strategy) and from each other. Figure 3 depicts the internal information flow within each robot. Since we do not force leader selection, the information on which leader should be selected is computed and displayed to the pilot through a pilot user interface.

The controllers used to control the UAVs are defined as follows: Assume the leader is  $l \in \{1, 2, 3\}$ . We define the dynamics of the subsystem as  $\dot{x} = f_l(x)$ , where  $l = 1, 2, 3$  as:

$$\begin{aligned} \dot{x}_l(t) &= p(t) - x_l(t) \\ \dot{x}_i(t) &= \sum_{j \neq i} (||x_i(t) - x_j(t)|| - k)(x_j(t) - x_i(t)), \\ & \quad i = \{1, 2, 3\} \setminus \{l\}. \end{aligned}$$

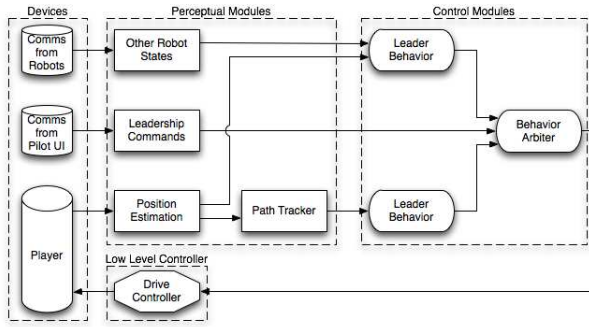


Fig. 3. Data flow architecture for each UAV agent.

This is a simple proportional compensator that drives the leader agent to the desired path, and executes weighted consensus dynamics on the follower agents so that each follower agent maintains a set distance of  $k$  with other agents. The result of this controller is a triangular formation, and the UAV team is shepherd by the leader vehicle.

As mentioned above, the pilot may choose to take remote control of the leader robot. In this mode of operation, the controllers of the UAV network can be described as follows.

$$\begin{aligned} \dot{x}_i(t) &= v(t) \\ \dot{x}_i(t) &= \sum_{j \neq i} (||x_i(t) - x_j(t)|| - k)(x_j(t) - x_i(t)), \\ i &= \{1, 2, 3\} \setminus \{I\}, \end{aligned}$$

where  $v(t)$  is the control input supplied by the pilot.

## B. Results

The system was tested in the open source Player/Gazebo simulation environment [6]. A hardware model for a UAV was implemented within Gazebo's open software environment. A fixed-wing aircraft was modelled, with 6 degrees of freedom, as shown in Figure 1. Furthermore, a graphical user interface was designed for input from the pilot and feedback for the pilot. The interface displays the global positions of the UAVs, the role (leader or follower) each UAV is currently playing (by coloring the respective UAV's icon), and the leader as suggested by the optimization algorithm (again, by coloring the respective UAV's icon). The pilot may choose a UAV to act as the leader by clicking on the icon of the desired agent. The pilot may also remotely control the leader by using a virtual joystick provided in the lower right-hand corner of the interface.

Demonstrations of the above described system were organized to highlight 3 features. The ability of the pilot to designate a new leader for the formation was demonstrated. Then, the affordance for the pilot to remotely control the leader UAV and then return the formation to autonomous control was demonstrated. Finally, the decision support is provided for the pilot each time the UAV network is reconfigured, or controlled manually by the pilot. Figures 4 and 5 demonstrate the capabilities of the above described system.

Figure 4 demonstrates the pilot's ability to designate a leader UAV from the team of UAVs. Figure 4(a) depicts the

pilot interface showing a team of 3 UAVs navigating along a predefined route. The designated leader is in front (colored red), while the other two UAVs follow behind (colored green). The pilot selects one of the two follower UAVs to become the new leader. The optimal control module suggests to switch back to the agent in front, which is marked in black as shown in Figure 4(b). This decision aid was not taken, and the selected UAV moves to take its place at the front of the formation, while the former leader moves to take its new place as a follower in the formation. Figure 4(c) shows the new formation with the newly designated leader in the front of the team.

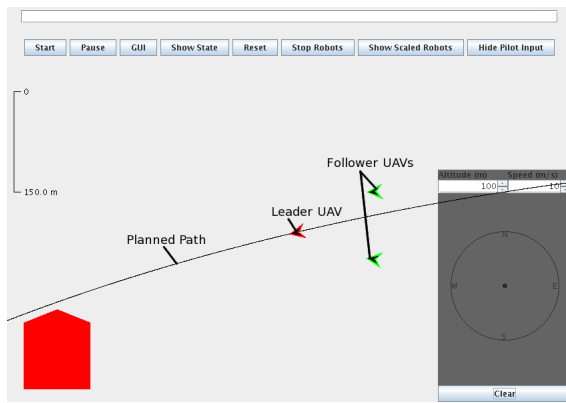
The affordance for the pilot to manually navigate the leader UAV is demonstrated by Figure 5. Figure 5(a) shows the formation of UAVs following their commanded route. An area of interest to the pilot (marked by the red polygon) is to the northwest of the formation. To explore the area of interest, the pilot uses the virtual joystick (shown in the bottom right corner of the interface) to pilot the leader UAV, as shown in Figure 5(b). The UAV under the pilot's control is marked blue. The other two UAVs continue to operate autonomously, albeit as followers. In this manner, the pilot is essentially in remote control of the entire formation. Once the pilot is satisfied with the exploration achieved, the leader UAV is returned to autonomous operation. Figure 5(c) shows the fully autonomous formation returning to its mission. Note that a suggestion was indicated by the optimal control module to switch to a follower agent while returning to the mission.

## V. CONCLUSION

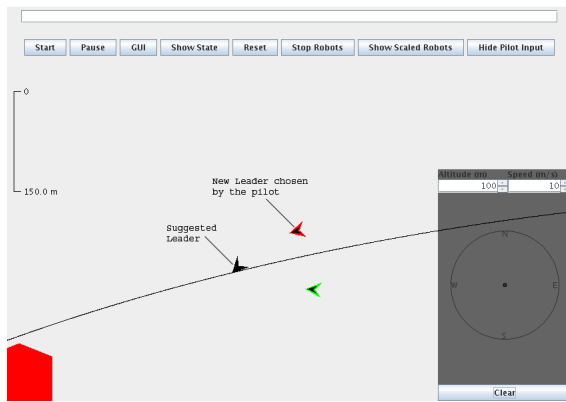
In this document, we report on our findings pertaining to pilot decision support for multiple unmanned aerial vehicles. In particular, we let the pilot interact with the system by selecting which of the agents should take on the role of a leader and a novel decision support feedback mechanism is provided. This feedback mechanism gives the operator access to the relative merits of selecting different vehicles as leaders, based on a novel, receding horizon approach to real-time optimal timing control for hybrid systems. Simulations in a realistic 3D environment support the soundness of the proposed approach.

## REFERENCES

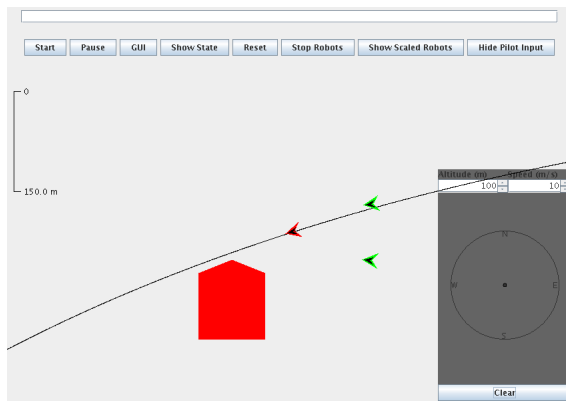
- [1] R.C. Arkin. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, Vol.8 No. 4. August 1989, pp 92-112
- [2] R.C. Arkin, Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, April 1992.
- [3] R.C. Arkin. Reactive control as a substrate for telerobotic systems. *Aerospace and Electronic Systems Magazine*, IEEE Volume 6, Issue 6, June 1991 Page(s):24-31
- [4] R.C. Arkin and K. Ali. Integration of reactive and telerobotic control in multi-agent robotic systems. *From animals to animats 3: Proc. Third International Conference on Simulation of Adaptive Behavior*, pages 473-478, 1994.
- [5] H. Axelsson, A. Muhammad, and M. Egerstedt. Autonomous Formation Switching for Multiple, Mobile Robots. *IFAC Conference on Analysis and Design of Hybrid Systems*, Sant-Malo, Brittany, France, June 2003.



(a) The formation of UAVs follow the planned path.

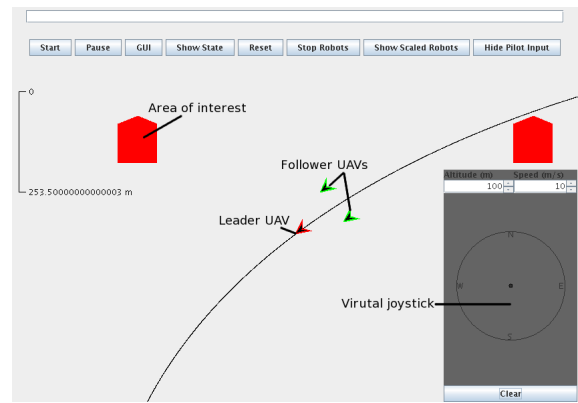


(b) The pilot chooses a new leader.

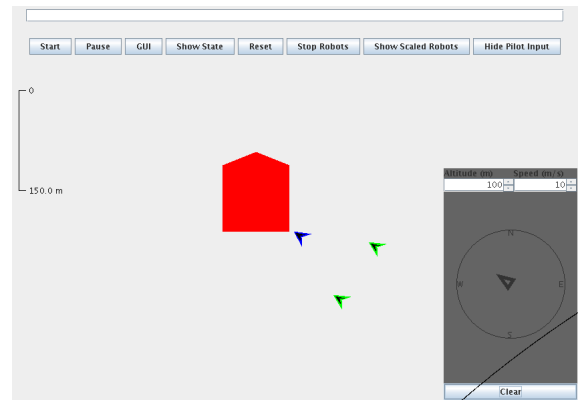


(c) The new leader takes its position at the front of the formation.

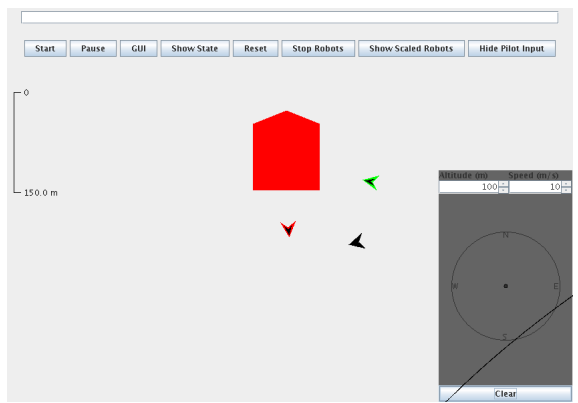
Fig. 4. Demonstration of the pilot designating a new leader for the formation of UAVs.



(a) The UAVs move along the planned path. An area of interest to the pilot lies to the northwest of the formation.



(b) The pilot remote controls the leader to the area of interest.



(c) The pilot returns the leader to autonomous control. The formation moves back to the planned path.

Fig. 5. Demonstration of the pilot exploring an area of interest by remote controlling the leader UAV.

[6] T.H.J. Collett, B.A. MacDonald, and B.P. Gerkey. "Player 2.0: Toward a Practical Robot Programming Framework". In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, Sydney, Australia, December 2005.

[7] M.L. Cummings. Human Supervisory Control of Swarming Networks, *2nd Annual Swarming: Autonomous Intelligent Networked Systems Conference*, June 2004.

[8] J.P. Desai, J. Ostrowski, and V. Kumar. Controlling Formations of Multiple Mobile Robots. *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, May 1998.

[9] M. Egerstedt, S. Azuma, and Y. Wardi. Optimal Timing Control of Switched Linear Systems Based on Partial Information. *Nonlinear Analysis: Theory, Methods & Applications*, 2006.

[10] M. Egerstedt, Y. Wardi, and H. Axelsson. Transition-Time Optimization for Switched-Mode Dynamical Systems. *IEEE Trans. on*

*Automatic Control*, Vol. AC-51, pp. 110-115, 2006.

[11] J. McLurkin, et. al. Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots, *AAAI Spring Symposium*, March 28, 2006

[12] Y. Wardi, X. Ding, M. Egerstedt, and S. Azuma. On-Line Optimization of Switched-Mode Systems: Algorithms and Convergence Properties. *IEEE Conference on Decision and Control, New Orleans, LA, Dec. 2007.*