

IMECE2005-80291

CONTROLLED VEHICLE EXCHANGE AND ALLOCATION IN DYNAMIC TEAMS

JUSTIN A. E. RUTHS
Columbia University
Department of Mechanical Engineering
500 West 120th Street
New York, New York 10027
United States
jar2131@columbia.edu

ANOUCK R. GIRARD
Columbia University
Department of Mechanical Engineering
500 West 120th Street
New York, New York 10027
United States
212-854-2962 Fax 212-854-3304
ag2363@columbia.edu

JOÃO BORGES DE SOUSA
Departamento de Engenharia Electrotécnica e Computadores
Faculdade de Engenharia de Universidade do Porto, Rua Dr. Roberto Frias
4200-465 Porto, Portugal
jtasso@fe.up.pt

ABSTRACT

Here, we present a solution for safe switching amongst dynamic teams of autonomous vehicles. By team, we mean a collection of vehicles that have a common mission, or objective. The design of our solution involves a distributed hierarchical control architecture. It contains five levels that successively abstract the motion of a single vehicle into maneuvers, coordinated maneuvers, tasks and team objectives. We do this in the framework of dynamic networks of hybrid automata and sliding mode control. We establish an information interchange protocol that minimizes bandwidth use and ensures robustness in the switching process, yet allows for communications at all levels of the control architecture. We present results from simulations to demonstrate and implement these ideas.

Keywords: switched systems, autonomous vehicles, dynamic teams

INTRODUCTION

There is large scale interest, especially in the military, in the safe and efficient use of unmanned vehicles (UVs). Currently many pilots and other trained staff are needed to control a single UV, which is extremely inefficient. One of our objectives is to invert the ratio of operators to vehicles to yield one human operator overseeing many autonomous vehicles. A team organization for a group of multiple vehicles is both intuitive and effective. Dynamic teams are more effective in

adapting to an evolving world. One important aspect of team organization and control concerns the allocation of vehicles to teams and the dynamic exchange of vehicles amongst different teams. The problem of preserving the properties of each team, and of their joint operations, under controlled vehicle exchanges is not a trivial matter.

The paper is organized as follows. In the second section, we define the domain of our problem and highlight two particular applications of our solution. Continuing with the second section, we give motivation for and description of the hierarchical control organization. Our third section focuses on the difficulties involved in switched systems and presents solutions for these problems. The fourth section introduces and describes the setup of and results from our simulation. We finish in section five with conclusions and future work.

LITERATURE REVIEW

A large body of research has been published in recent years about motion control of Unmanned Aerial Vehicles (UAVs). An increasingly large number of hardware platforms [9-13] are available for those wishing to validate their ideas experimentally. At this point, much of the work available in the literature [14-15] deals with motion control of vehicles, usually in the form of waypoint following or trajectory tracking. Parallel advances in wireless communication technologies are enabling applications that include cooperative control of multiple UAVs working together to accomplish a greater mission goal. Multiple-UAV control strategies are emerging,

for example in battlefield scenarios where N UAVs are assigned to strike T known targets in the presence of dynamic threats [16, 17] and in the fields of synchronized path planning and cooperative rendezvous problems where multiple UAVs must arrive at their targets simultaneously [7, 16, 18-20]. Cooperative strategies have also been considered, for example in [21], which considers cooperative search strategies under collision avoidance and communication range constraints, and in [22], which presents a completely decentralized, hybrid systems approach and does not require that the UAVs stay within communication range of each other, as well as in [24, 25] which consider formation flight problems. Other approaches to supervision and control of multiple UAVs are considered in [23, 26]. However, the problems of dynamic teams, whose composition may change at run time, and of team supervision in this context, have not been addressed to date. In this paper, we attempt to present a framework for the coordinated operation of multiple UAVs, with switching of vehicles between teams during missions.

The current literature on switched control systems is reviewed in Liberzon and Morse's landmark paper [1]. In this problem setup there is a plant, a set of controllers, and a switching block to select the controller which controls the plant at any given time. They give a thorough overview of techniques used to find either a common Lyapunov function or a switching function that will stabilize the system. The authors have treatments for arbitrary switching of stable systems as well as for more limited cases.

Controlled vehicle exchange and allocation in dynamic teams poses other types of switching control problems which occur in a control architecture. Papers [2-4] give background on the usefulness and techniques for implementing a hierarchical control architecture. In addition, elements of [3] include attention to protocol design. As we motivate the difficulties associated with higher level switching, [5] provides us with methodologies to smooth the transitions. The authors discuss a transition management pattern that incorporates aspects such as blending functions. The sliding mode controller used in both simulations is drawn from [6].

ARCHITECTURE AND ORGANIZATION

Problem Statement

The atomic unit for battlefield and battlefield-like environments is the human soldier – which we have replaced with an autonomous vehicle. We seek to create useful ways to organize these atomic units to enable battlefield and battlefield-like coordination. To this end, we create the notion of a team. We define a team, as stated in the abstract, as a grouping of n vehicles which have a common mission, where n is unity or greater. Note that it is possible for a team to be composed of only one vehicle. A subset of the vehicles in a team can form a subteam within the original team to enable even more complex coordination. However, this property is not explored in this paper. Teams have other invariant properties that will be discussed in later sections.

If teams were to remain intact and unmodified throughout the course of a battlefield-type situation, they would themselves be considered fundamental, or indivisible, units. However, we compose a team such that it can be modified and adjusted. We can imagine many situations in which vehicles must be reallocated during the event, to adjust to changing external parameters, to maximize the effectiveness of the mission, or to maintain the safety of the units involved. Vehicles must be withdrawn from teams as they return for fuel, servicing, or are destroyed. Vehicles need to be added to teams once they are refueled, serviced, or replaced. Finally, vehicles must be able to change teams and form new teams so as to reallocate resources or share a scarce resource. All of these problems fall within the problem domain of switched systems.

Example Scenario

We consider two specific scenarios to exemplify the use of switching. Consider, first, two teams of three vehicles each. Team 1 has vehicles V1, V2, and V3; Team 2 has vehicles V4, V5, and V6. The geographic region of interest can be divided into three sectors. Team 1 patrols sector A; Team 2 patrols sector C; the intermediate area, sector B, is left open. Seeking full coverage of the region, a new team, Team 3, is created and assigned to sector B. The regional supervisor requests one robot from both Team 1 and Team 2 to join Team 3. Vehicle V3 from Team 1 and V6 from Team 2 are switched to Team 3. Accomplishing this double switching scenario under limited bandwidth and with the threat of communication dead zones is not trivial.

The second situation we will consider is one of Battlefield Damage Assessment (BDA). Targets are identified for bombing and divided into two sectors. Bombing Team 1 is assigned to sector A and Bombing Team 2 is assigned to sector B. Bombers are relatively plentiful and readily available; therefore, it is possible to have several if not many of these aircraft. The necessary follow-up to bombing a target is to verify that the target is destroyed. Planes with this capability have expensive video surveillance equipment and are a rare resource. Due to the lack of availability and high cost, such planes – we will call them classifiers – are shared between the targets of the two sectors. As targets are destroyed, the classifier is called to verify the successful bombing. For purposes such as avoiding collisions and ensuring communication, when the classifier changes geographic sectors it is advantageous for the classifier to also switch teams. The effect is a single vehicle that repeatedly switches between two teams.

Hierarchical Control Organization

Our five level hierarchy of control organization, shown in Fig. 1, is both functionally motivated and suggestive of the chain of command that is typically found in military situations. The top level deals with large scale and high level decisions. Each successive level reduces the scale of its scope and increases its direct access to the dynamics of the vehicles

themselves. This organization includes the idea of teams within its construction. Our solution is for the general case of switching, thus the intended missions and objectives of the levels are not particular to any exact application.

The levels in Fig. 1, from top to bottom, can be described as follows:

Layer 5: Regional Supervisor. The regional supervisor is an overseeing entity in charge of a geographic region (generally large). Directly linked beneath the regional supervisor are the team supervisors. The regional supervisor generates objectives for the teams to follow. Oftentimes a human operator is the regional supervisor, or capable of directly intervening in the processes of an autonomous agent.

Layer 4: Team Supervisor. The team supervisor receives objectives from the regional supervisor and intelligently configures the supervised team to accomplish the given objective. Directly beneath the team supervisor level are the vehicle supervisors. The team supervisor parses the objectives from the regional supervisor into maneuvers for the vehicle supervisors. The team supervisor is responsible for adapting the present resources of its team to fulfill the requirements of the regional supervisor's requests. Higher level communication between members of a team is done through the team supervisor.

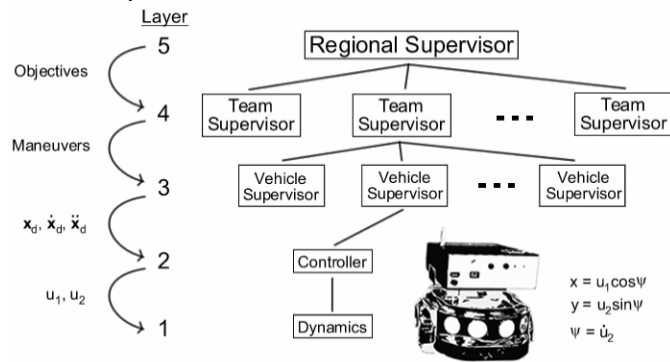


Figure 1. Hierarchical Control Architecture

Layer 3: Vehicle Supervisor. There is a unique vehicle supervisor associated with each vehicle. This level manages all aspects of the vehicle itself. Directly linked to the vehicle supervisor is the controller and motion planning level. The vehicle supervisor takes maneuvers from the team supervisor and supplies the controller with waypoints (points with corresponding desired position, velocity, and acceleration). Team communication of vehicle resources is done at this level.

Layer 2: Controller/Motion Planning. The controller is responsible for planning the inputs to the vehicle dynamics with desired waypoints from the vehicle supervisor. Object and collision avoidance is also part of the vehicle supervisor. The formation control aspects of team communication are done at the controller level.

Layer 1: Vehicle Dynamics/Autopilot. Given commands from the controller, the vehicle hardware and onboard autopilot

routines drive the dynamics of the vehicle. Vehicle monitoring and adjustments are made at this level.

Level 5 is most often located away from the geographic region and is contacted remotely. Level 4 is generally also off location, however, it can be incorporated into one of the vehicles in the team. Flexibility is lost by locating the team supervisor in a particular vehicle; there is also an increased chance of failure in this configuration. Level 2 and 3 can be accomplished off the vehicle though it is more effective to have it housed onboard. Level 1 is generally required to be on the vehicle.

Information Exchange between Levels

Due to the differing scopes and responsibilities at the different levels of the control hierarchy, there are unique information exchanges that occur between the levels. As an example of the communication, we will take the particular case of a team needing to patrol around a city. The information exchange is broken down into the following levels:

Objectives: Objectives are passed from the regional supervisor (level 5 in Fig. 1) to the team supervisor (level 4). These are high level general orders. In our particular case, the objective might be to patrol in a circle around a city, with parameters for the center and radius of the circle. Additional parameters, such as speed/frequency or passive/aggressive, can also be dictated from this level.

Maneuvers: Maneuvers are passed from the team supervisor (level 4) to the vehicle supervisors (level 3). While still general commands, they are more specific than the objective from which they are generated. For our patrolling team, the team supervisor would give incremental maneuvers to vehicle supervisor level –taking them around the path of the circle. Here we would use a simple goto-maneuver –consisting of a desired position, velocity, and acceleration. By placing these goto-maneuvers in succession, the team supervisor is then able to direct the team through the patrol path.

The controller, Level 2, is able to extract the desired position, velocity, and acceleration in order to generate the commands for the vehicle dynamics.

SWITCHING

The difficulties in switched systems occur during the switching. The domain of our problem is such that if we could have two robots, one on the left and one on the right, close our eyes, and when we open them again there be one robot on the left and two on the right our investigation would be inconsequential. However, what happens between the initial and final configuration of teams is not trivial and takes great care and attention for it to take place smoothly. In our pursuit for a solution we consider general difficulties that can be emulated in our simulations. This being a large enough collection of items, we will not consider the individual happenstances such as mechanical failure, although when

abstracted these can usually be considered one of our more general cases.

Problems in switching occur at all levels of the hierarchy. Low level switching issues considered in [5] include aspect such as discontinuous jumps in the state of the controller. At higher levels a communication blackout between team supervisors during the switch of a vehicle can lead to very unpredictable behavior. If connection is reestablished, synchronizing the vehicle again with the team is not necessarily a straightforward problem either.

Distributed control framework for dynamic teams of networked vehicles and sensor systems

The distributed control framework has a graph structure where the nodes represent team controllers and the arcs represent the communication links. The state of the distributed control framework is termed a configuration of the system, and is described by this graph and the state of each controller. The configuration of the system may change with time to adapt to changes in the environment, in the control objectives, and in the controlled assets. The evolution of the control framework follows organizational principles which govern the switches among different configurations.

The distributed control framework is layered in the sense that the controllers are nested. This leads to a structure with adjustable layering. A team controller and its controlled dependants (vehicles, sensors, and other controllers) form a team. A team delivers properties which cannot be delivered by any of its constituent elements. The nested structure of controllers leads to nested sub-teams within a team.

The vehicle/sensor to team allocation is dynamic: vehicles and sensors come and go. For inter-operability, vehicles and sensors have a well-defined high-level command set.

We now introduce some terminology for use in a mathematical description of vehicles and their capabilities. Vehicle is a generic term for entities with motion capabilities of different types such as trucks, UAVs, etc. There is a finite set of types, called $VehicleTypes = \{bomber, classifier\}$. The classifier type vehicle is not armed and is used only for BDA. Each vehicle is able to execute a set of maneuver types and provide a set of services. A maneuver is a parameterized pattern of behavior, with phased operations, which is modeled as a hybrid automaton. There is a finite set of types of maneuvers, called $ManeuverTypes = \{goto, attacktarget, loiter, followpath, inspect, jointeam\}$, and a finite set of services, called $ServiceTypes = \{inspect, jamlocation\}$. The maneuver types are as follows: *goto* consists of reaching a given destination; *attacktarget* of attacking a target using a given tactic; *loiter* of staying close to a given point; *followpath* of tracking a given path; *inspect* of moving to inspect a given location using a given tactic; and *jointeam* consists of moving to a given region to join a team.

The function $maneuver(p): VehicleTypes \rightarrow 2^{ManeuverTypes}$ gives the set of maneuvers for a vehicle of type p ; the function

service is defined similarly. Maneuver types are parameterized to allow the concurrent delivery of services.

Modeling framework

We describe the model of the distributed control structure. We do this in the framework of dynamic networks of hybrid automata (DNHA), which is briefly described in the sequel. A hybrid automaton is $H = (L, D, E)$ where: L is a set of control locations; $D : L \rightarrow ODEs$ where $D(l)$ is the controlled differential equation at location l ; $E \subseteq \{L \times Guard \times Event \times Action \times L\}$ are the edges, where an edge $e = (l, g, v, j, m) \in E$ is an edge from location from l to m with guard g , event v , and action j . The state of a hybrid automaton is a pair (l, x) where l is the control location and $x \in \mathbb{R}^n$ is the continuous state. Informally, DNHA allow for interacting automata to create and destroy links among themselves and for the creation and destruction of automata. Formally, this requires the following extensions to the hybrid automaton model: hybrid automata have types and there are instances of these types; the state x also includes link variables to other hybrid automata; the Actions include creation and deletion of hybrid automata. Link variables enable the control actions to refer to the state of other hybrid automata.

Vehicle control structure

The vehicle control structure is composed of the vehicle supervisor, the mission supervisor, and a maneuver controller (there is a maneuver controller for each type of maneuver).

The vehicle supervisor is a hybrid automaton VS , where $L = \{Exec, Error, Idle\}$, the state is $x = \{motionvariables, teamcontroller, regionalsupervisor, maneuvercontroller, missionsupervisor, servicesdelivered, free\}$, *Event* includes the commands accepted by the vehicle, *Action* includes the commands to create maneuver controllers and configuration commands. We briefly describe the transition structure E . In the *Idle* state, the supervisor accepts a maneuver command from the *teamcontroller* or from the *missionsupervisor* and takes the transition to *Exec*. On this transition, it creates a maneuver controller and sets the variable *maneuvercontroller* to it. The transition from *Exec* to *Idle* is taken when an abort command is received from the *teamcontroller* or from the *missionsupervisor*, or when the *maneuvercontroller* completes the execution of the current maneuver.

Each vehicle operates under the control of either its mission supervisor or of a team controller. In the first case it operates independently, in the second one as part of a team. In both cases the vehicle executes one maneuver at a time, while providing services. In the first case, the vehicle supervisor is linked to the mission supervisor; in the second case the vehicle supervisor is linked to an external controller. The interaction protocol for the two types of links is the same: the vehicle supervisor receives commands to execute maneuvers, as well as configuration commands, and sends back termination or error messages.

Now we discuss the properties of the maneuver controllers required to execute missions such as the ones described in the previous section and sketch the structure of the *jointeam* maneuver which encodes the protocol required for a classifier vehicle to join a team. The *jointeam* maneuver is a hybrid automaton MJ, where $L = \{\text{independent, coordinated, stop}\}$. The state is $x = \{\text{team, reach}\}$, where *team* is a variable which is set to the identifier of the team supervisor, and *reach* is a Boolean variable which is set to TRUE when the vehicle which is executing the maneuver is able to contact the team supervisor. We briefly describe the transition structure E. Execution starts in the *independent* state where the Boolean variable *reach* is set to FALSE. The transition to *coordinated* is taken when *reach* is set to TRUE, which means that the vehicle is able to communicate with the team supervisor. In this state it executes a coordination protocol to join the team. We omit the details of this protocol. Finally, if join operation succeeds the transition to *stop* is taken, the variable *teamcontroller* in the vehicle supervisor is set to the current value of *team*, and the maneuver terminates.

The variable *regionalsupervisor* is permanently bound to the Regional Supervisor described in the previous section – we assume that each vehicle is in direct communication with the regional supervisor. The variable *teamcontroller* is set to a particular team controller or to nil by commands from the *regionalsupervisor*. This is explained next.

Team controller

The team controller is a hybrid automaton TS, where $L = \{\text{coordinated, uncoordinated, Idle}\}$, the state is $x = \{\text{bomberteam, classifierteam, regionalsupervisor, stateofplan, request, pendingrequest, requestingcontroller, freeclassifier, plan}\}$, where *bomberteam* and *classifierteam* are the sub-teams of *bomber* and *classifier* vehicles, *plan* is a hybrid automaton which encodes the phased operations for the vehicles in the team, *stateofplan* is the current state of execution of the plan, *request* is a Boolean variable which is set to TRUE when there is an internal pending request for a classifier vehicle, *pendingrequest* is a Boolean variable which is set to TRUE by the Regional Supervisor when there is an external pending request for the classifier vehicle, *requestingcontroller* is the name of a team controller requesting for a classifier vehicle, and *freeclassifier* is a Boolean variable which is set to TRUE when classifier type vehicles are no longer needed for the execution of the plan. Basically, the variables *bomberteam* and *classifierteam* encode links to the vehicle supervisors of both sub-teams. In the coordinated state both the *bomberteam* and *classifierteam* are non-empty and their motions are coordinated. In the uncoordinated state the *classifierteam* may be empty. *Event* includes requests of a classifier type vehicle to join the team, when it is located within communication range. *Action* includes commands from the Regional Supervisor to remove from the team a set of vehicles, namely of classifier type, and requests for a classifier type vehicle sent to the Regional Supervisor. We briefly describe the

transition structure E. It basically consists of plan execution. To each state of the plan there corresponds a state of execution. The transition from *uncoordinated* to *coordinated* takes place when the current state of *plan* is over and *classifierteam* is not empty. Otherwise a transition to *Idle* takes place, the *request* variable is set to TRUE, and a request for a classifier vehicle is sent to the Regional Supervisor. Then execution pauses and the vehicles are commanded to loiter in pre-assigned positions. The transition from *Idle* to *coordinated* takes place when a classifier vehicle joins the team and *classifierteam* is set to that vehicle. In both the *coordinated* and *uncoordinated* states both sub-teams execute the actions encoded in the plan. There are two self-transitions in the *uncoordinated* state. The first one takes place when *pendingrequest* is TRUE and *classifierteam* is not empty; on this transition the team controller removes the classifier vehicle from *classifierteam*, sets the *teamcontroller* variable in the classifier vehicle to *nil*, and commands this vehicle to execute a join maneuver to join the requesting team controller. The second one takes place when *freeclassifier* becomes TRUE; on this transition the team controller removes the classifier vehicle from *classifierteam*, sets the *teamcontroller* variable in the classifier vehicle to *nil*, and sets the free vehicle in its vehicle supervisor to TRUE. The plan is a nested hybrid automaton. To each node there corresponds a state of execution which is modeled as another hybrid automaton. Space limitations preclude a detailed description.

The team controller has the following properties

1. Plan execution proceeds until termination or is paused when a classifier type vehicle is requested;
2. Plan execution is resumed when a classifier type vehicle joins the team.
3. Transfers a classifier type vehicle to the Regional Supervisor upon reception of a request from it, and after completion of the current stage of the plan.
4. Transfers a classifier type vehicle to the Regional Supervisor when the *freeclassifier* variable is set to TRUE, meaning that the vehicle is no longer required by the team.
5. Does not block if the plan does not block and the maneuvers terminate as specified.

Configurations and organization principles

In what follows, and for the sake of clarity, we consider the following assumptions:

1. There is one Regional Supervisor and two team supervisors.
2. The initial allocation of bombers to team supervisors does not change.
3. There is one classifier vehicle which is shared between the two team supervisors.
4. Both team supervisors require the intervention of a classifier vehicle at a certain point in the execution of their respective plans.
5. The two teams are not capable of communicating between them.

In this case the classifier vehicle has to switch between the two teams. Moreover, it may be required to leave one team and to operate independently until it reaches communication range of the other team. When this happens it requests permission to join the team.

In the normal operation the distributed control structure system is in one of the following configurations: a) two-team; b) shared-team; c) three-team. In the first configuration, the classifier vehicle is part of one of the teams. In the second configuration, the classifier vehicle is operating independently after leaving one team to join the other team. In the third configuration the classifier vehicle is no longer required by both teams and is acting as a free agent. The first configuration is formally described next. The other configurations are defined in a similar fashion. Let the identifier of the Regional Supervisor be ReS. The first configuration satisfies the following property:

$$\forall v \in \text{TeamControllers}, \text{regionalsupervisor}(v)=\text{ReS} \wedge \text{free}(v)=\text{FALSE} \wedge ((\forall s \in \text{bomberteam}(v): \text{teamcontroller}(s)=v) \wedge (\forall t \in \text{classifierteam}(v): \text{teamcontroller}(t)=v))$$

At the configuration level the control problem consists of ensuring the system does not block and keeps transitioning between the first two configurations as far as there is a need to share the classifier vehicle, and that the system transitions to the third configuration when the classifier vehicle is no longer needed.

Regional supervisor and coordinated operations

The Regional Supervisor is a hybrid automaton RS, where $L = \{\text{two-team}, \text{shared-team}, \text{three-team}\}$, the state is $x = \{\text{request}, \text{requestingteam}, \text{teams}, \text{freevehicle}\}$, where *request* is a Boolean variable which is set to TRUE when there is a pending request from one team for a classifier vehicle, *requestingteam* is set to the team controller which is requesting a classifier vehicle, *teams* is the set of links to the team controllers under its control, and *freevehicle* is the set of vehicles which are not engaged in the operations of the other teams. There are several possibilities for the transition E. We describe one next. The states describe the configuration of the system. A transition from *two-team* to *shared-team* takes place when *request* is TRUE and the team controller in control of the classifier vehicle relinquishes control over it. On this transition *request* is set to FALSE and *requestingteam* to *nil*. The transition from *shared-team* to *two-team* takes place when the classifier vehicle, which is executing the *jointeam* maneuver joins the other team. There is a self-transition in the *two-team* state. When a request for a classifier vehicle is received from one team the *request* and *requestingteam* variables are updated accordingly and the Regional Supervisor forwards the request to the other team by setting the corresponding variables. The transition from *two-team* to *three-team* takes place when the classifier vehicle is released by its team controller and is added to the *freevehicle* set.

The Regional Supervisor has the following properties

1. Forwards requests for a classifier vehicle to the team where it belongs.
2. Upon receiving a classifier type vehicle from one team assigns it to the other one if the corresponding *freeclassifier* variable is FALSE.
3. Upon receiving a classifier type vehicle from one team assigns it the free agent status if the *freeclassifier* variables from both teams are TRUE.

The correct operation of the distributed control structure is ensured by the execution properties of the Regional Supervisor, Team Controllers, and Maneuver Controllers, in particular the *jointeam* maneuver.

SIMULATION AND RESULTS

To fully demonstrate the effectiveness and feasibility of our approach we construct simulations to match the real world problems. Both scenarios explained earlier in the paper will be considered. In order to establish a robust solution, careful consideration must be taken in the communication protocol between teams and between levels.

Protocol

The design of the communication protocol is an important facet of the architecture for the solution. In operating conditions, the availability and bandwidth of communication is often a limiting factor. Our protocol is constructed to be robust against connection failures due to dead zones. It also takes into account situations in which two teams ask for the same robot at the same time. These are examples of issues that can break a simply designed solution.

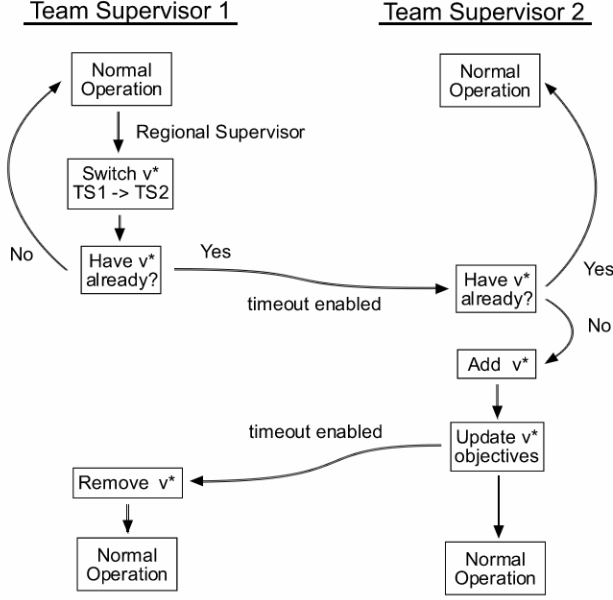


Figure 2. Switching Protocol

The flow chart of Fig. 2 diagrams the formal interchange between teams that takes place to switch a vehicle. The figure illustrates one instance of robust protocol design within our solution. Using this series of checks, we ensure that the switched vehicle is never left without a team supervisor or without an objective. For instance, both exchanges between team supervisors have timeout features. If communication is lost between team supervisors or the link to the switched vehicle is lost during this part of the protocol and not reestablished within the timeout period, then the states of the teams and vehicles return to their original values.

Another feature to circumvent the effects of communication failure, both vehicle supervisor and team supervisor have a copy of the team's objective (which can be represented with a series of maneuvers). When a maneuver is completed by the vehicles in a team, and there is communication between the vehicles and the team supervisor, the objective, or maneuver list, is synchronized. If a team supervisor loses communication with the vehicles and at the same time receives a new vehicle into its command, it can issue the last synchronized objective that it has. Once communication is reestablished, any discrepancy can be eliminated.

Simulation Design

The first simulation is done using vehicles that are modeled with non-holonomic kinematics. The vehicles are given maximum linear and turning velocities as well as a minimum linear velocity of zero. The UGVs start at rest using a simple proportional controller and once a speed is established

they switch to the sliding mode controller discussed in the following section.

The second, BDA, simulation is constructed the same, except that the vehicles are given finite positive minimum velocities – thus resembling UAVs. These vehicles start with an initial velocity and use the sliding mode controller throughout the simulation.

Controller

We consider the design of a trajectory tracking controller for small, non-holonomic ground robots. We use a kinematic model to represent the robot.

$$\begin{cases} \dot{x} = u_1 \cos \psi \\ \dot{y} = u_1 \sin \psi \\ \dot{\psi} = u_2 \end{cases} \quad (1)$$

The directions x and y are in a frame that is fixed with respect to the ground. The control variables are u_1 and u_2 , the forward speed and turn rate.

Our controller design is based on sliding mode control [9] and dynamic extension [8]. We start by defining our position vector, η , and our desired position vector, η_d .

$$\eta = \begin{pmatrix} x \\ y \end{pmatrix} \quad (2)$$

$$\eta_d = \begin{pmatrix} x_d \\ y_d \end{pmatrix} \quad (3)$$

We define e to be the position error.

$$e = \eta - \eta_d \quad (4)$$

We select our sliding surface to be:

$$S = \dot{e} + Ke \quad (5)$$

If S goes to zero, then we have exponentially decreasing error dynamics, at a rate set by K , which is a positive definite matrix that we select as a control parameter.

$$S \rightarrow 0 \Rightarrow \dot{e} = -Ke \Rightarrow e \rightarrow 0 \quad (6)$$

Rewriting the equation for S , we have:

$$S = \dot{\eta} - \dot{\eta}_d + Ke \quad (7)$$

Defining some notation, we set: $V = \dot{\eta}$ and $V_R = \dot{\eta}_d - Ke$.

We then have:

$$S = V - V_R \quad (8)$$

We now need to focus on finding a strategy to make the velocity, V go to V_R . If $V=V_R$, then S equals zero, which drives

the position error to zero, which is our goal. We will use a strategy called dynamic extension to that aim. We have:

$$S = \begin{pmatrix} u_1 \cos \psi \\ u_1 \sin \psi \end{pmatrix} - \dot{\eta}_d + Ke \quad (9)$$

We take a derivative of our sliding surface, S.

$$\dot{S} = \begin{pmatrix} \dot{u}_1 \cos \psi - u_1 \sin \psi \dot{\psi} \\ \dot{u}_1 \sin \psi + u_1 \cos \psi \dot{\psi} \end{pmatrix} - \ddot{\eta}_d + K\dot{e} \quad (10)$$

Rewriting the above equation in matrix form, and remembering that $\dot{\psi} = u_2$:

$$\dot{S} = \begin{bmatrix} \cos \psi & -\sin \psi u_1 \\ \sin \psi & \cos \psi u_1 \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ u_2 \end{bmatrix} - \ddot{\eta}_d + K\dot{e} \quad (11)$$

Setting some notation:

$$A = \begin{bmatrix} \cos \psi & -\sin \psi u_1 \\ \sin \psi & \cos \psi u_1 \end{bmatrix} \quad (12)$$

$$v = \begin{bmatrix} \dot{u}_1 \\ u_2 \end{bmatrix} \quad (13)$$

Notice that the A matrix is invertible for $u_1 \neq 0$. We then have:

$$\dot{S} = Av - \ddot{\eta}_d + K\dot{e} \quad (14)$$

We would like to have exponential dynamics for our sliding surface: $\dot{S} = -\Lambda S$ where Λ is a positive definite matrix. Our desired behavior is then:

$$\dot{S} = Av - \ddot{\eta}_d + K\dot{e} = -\Lambda S \quad (15)$$

We can then get the vector v from the equation:

$$v = A^{-1}[\ddot{\eta}_d - K\dot{e} - \Lambda S] \quad (16)$$

The vector v gives us values for u_2 and \dot{u}_1 . What we really need are values for u_1 and u_2 . We pass the v vector through a system that integrates the first field to obtain the required values. This is referred to as dynamic extension.

Results

In our simulation of dynamically creating a new team (Fig. 3), two teams are given directives to patrol two sectors separated by a gap. Referring back to the description of our example scenarios given above, these first two teams are to patrol sectors A and C (left and right respectively). During their patrol, a third team is created to patrol sector B (the center gap between A and C). One robot is requested to join the third team from each of the first two teams. By patrolling, here we mean alternating between two waypoints.

If the robots were allowed an infinitely fast turning rate, we would expect their trajectories to be lines connecting the two waypoints. Instead, the shape of these vehicles' paths is

due to their finite maximum turning rate. A larger maximum would make the arcs smaller in radius. The sharp corners are present because the vehicles slow to a stop once reaching the waypoint and then begin on the next segment. At such a slow speed the "corner" can seem discontinuous, however, it is a function of the slow speed and the on-point turning capability of the vehicles. Notice that all vehicles in this simulation run in loops counter-clockwise. We have also assembled the vehicles into a simple cross-shaped formation, which makes the trajectories slightly offset from one another.

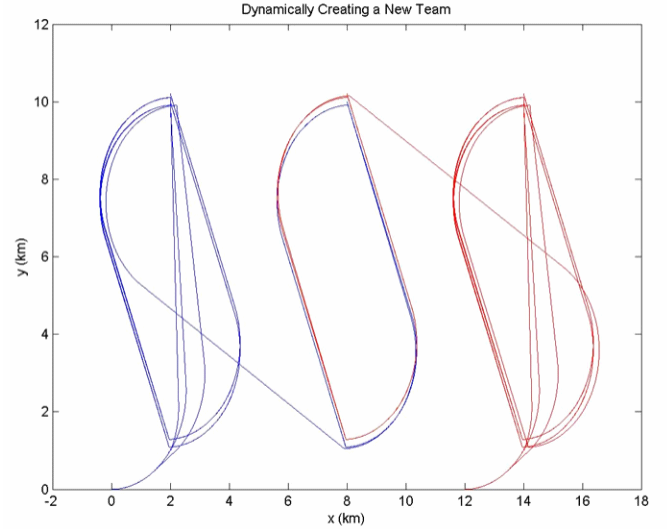


Figure 3. Dynamically creating a new team

The Battlefield Damage Assessment (BDA) simulation (Fig. 4) has two teams of bomber aircraft (denoted by dotted and dashed lines). For visual clarity, each team has one bomber, however, it is possible to have as many as is needed (see previous simulation for teams of multiple vehicles). The bombers are ordered to destroy targets, here arranged in a grid-like pattern and noted by 'x' symbols. The dotted bomber is given the left targets; the dashed bomber the right targets. Once a bomber has bombed a target, the classifier aircraft (solid line) is requested to certify a successful bombing. Until the classifier verifies the bombing, the bomber circles the target. When the classifier is requested from another team, the classifier switches teams. This behavior creates the hopping pattern seen in the classifier's trajectory. Notice that the dotted bomber always completes its runs before the dashed bomber – therefore the dashed bomber circles its target longer than the dotted bomber.

Battlefield Damage Assessment

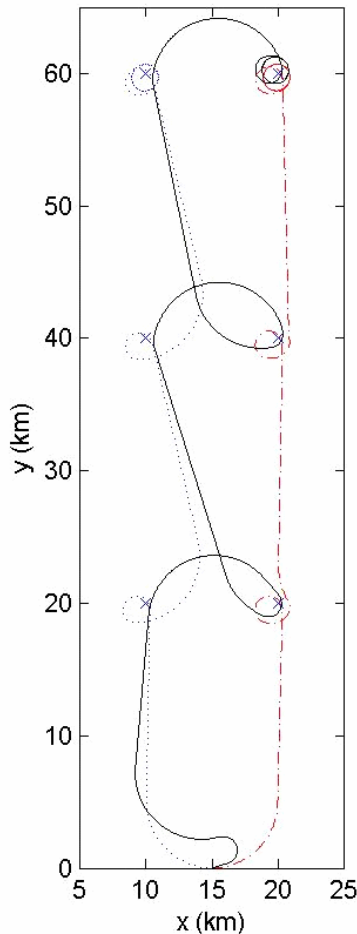


Figure 4. BDA Simulation

CONCLUDING REMARKS

The literature until now has been sparse in dealing with the practical aspects of switching in a battlefield-like scenario. In this paper we have proposed a hierarchical control structure and developed a protocol to deal with this kind of situation. Simulations, but also the methodology, lead us towards a direction of exploration and growth into this untapped area of autonomous control.

Future work will include careful consideration of problems in switching, especially in more complex scenarios. Switching teams between different regional supervisors, as for a traveling convoy, will take the switching issue to the next higher level on our hierarchical model. Finally we will incorporate this design into existing work from other areas to form a more complete and functional model of real world situations.

ACKNOWLEDGMENTS

The authors wish to thank the support of the Columbia University Internal Funds.

REFERENCES

- [1] Liberzon, D., Morse, A. S., 1999, "Basic problems in stability and design of switched systems," *IEEE Contr. Syst. Mag.*, **19**, pp. 59–70.
- [2] J. Borges de Sousa, Anouck R. Girard and J. K. Hedrick. "Elemental Maneuvers and Coordination Structures for Unmanned Air Vehicles". Proceedings of the 43rd IEEE Conference on Decision and Control, Paradise Island, The Bahamas, Dec 14-17, 2004.
- [3] A. Girard, J. Borges de Sousa and J.K. Hedrick. A Hierarchical Control Architecture for Mobile Offshore Bases. *Marine Structures Journal*, **13**, No. 4-5, July 2000, pp.459-476.
- [4] S.C. Spry, A.R. Girard and J.K. Hedrick. Convoy Protection using Multiple Unmanned Aerial Vehicles: Organization and Coordination. Accepted for Publication in the 2005 American Control Conference, ACC 2005.
- [5] Guler, Murat, Clements, Scott, Wills, Linda M., Heck, Bonnie S., Vachtsevanos, George J., 2003, "Transition Management for Reconfigurable Hybrid Control Systems". *IEEE Contr. Syst. Mag.*, **23**, pp. 36-49.
- [6] Girard, Anouck R., Spry, Stephen C., Hedrick, Karl, J., "Leaderless Formation Control Using Sliding Control and Dynamic Extension". Accepted for Publication at the Advanced Intelligent Mechatronics Conference, July 2005.
- [7] T. McLain and R. Beard, August 2000, "Cooperative Rendezvous of Multiple Unmanned Air Vehicles", AIAA Guidance, Navigation and Control Conference, Denver, CO, Paper no. AIAA-2000-4369:AAO-37126.
- [8] J.J.E. Slotine and W. Li, 1991, *Applied Nonlinear Control*, Prentice Hall.
- [9] J. S. Jang and C. Tomlin, August 2002, "Design and Implementation of a Low Cost, Hierarchical and Modular Avionics Architecture for the DragonFly UAVs", in the Proceedings of the AIAA Guidance, Navigation, and Control Conference, Monterey, AIAA Paper Number 2002-4465, 11 pages.
- [10] S. Bayraktar, G. Fainekos, and G. J. Pappas, December 2004, "Experimental Cooperative Control of Unmanned Aerial Vehicles", submitted to the Proceedings of the 43rd IEEE Conference on Decision and Control, The Bahamas.
- [11] D. H. Shim, H. J. Kim, H. Chung, S. Sastry, "A Flight Control System for Aerial Robots: Algorithms and Experiments", in Proceedings of the 15th IFAC World Congress on Automatic Control, July 2002.
- [12] V. Gavrillets, A. Shterenberg, M. Dehaleh and E. Feron, "Avionics System for a Small Unmanned Helicopter Performing Aggressive Maneuvers", in Proceedings of the 2000 Digital Avionics Systems Conference.

- [13] R. Hindman, R. Franz and J. Hauser, "The Virtual Ducted Fan: A Control Design and Test Platform for the Caltech Ducted Fan", Proceedings of the 40th IEEE Conference on Decision and Control, 2001.
- [14] M. van Nieuwstadt, R. M. Murray, "Real-Time Trajectory Generation for Differentially Flat Systems with Unstable Zero Dynamics", Proceedings of the IFAC 1996.
- [15] D. H. Shim, H. J. Kim, H. Chung, S. Sastry, "A Flight Control System for Aerial Robots: Algorithms and Experiments", 15th IFAC World Congress on Automatic Control, July 2002.
- [16] R.W. Beard, T.W. McLain, M. Goodrich and E.P. Anderson, "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles", IEEE International Conference on Robotics and Automation, Washington DC, May, 2002.
- [17] J. Borges de Sousa, T. Simsek and P. Varaiya, "Task Planning and Execution for UAV teams", Accepted for Publication, IEEE Conference on Decision and Control, Bahamas, December 2004.
- [18] J. Borges de Sousa, A.R. Girard and J.K. Hedrick, "Elemental Maneuvers and Coordination Structures for Unmanned Air Vehicles", Accepted for Publication, IEEE Conference on Decision and Control, Bahamas, December 2004.
- [19] T.W. McLain, P.R. Chandler, S. Rasmussen and M. Pachter, "Cooperative Control of UAV Rendezvous", Proceedings of the 2001 American Control Conference, **3**, pp. 2309 – 2314.
- [20] P.R. Chandler, M. Pachter, and S. Rasmussen, "UAV Cooperative Control", Proceedings of the 2001 American Control Conference, **1**, pp. 50-55.
- [21] R.W. Beard and T.W. McLain, "Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints", Proceedings of the 42nd IEEE Conference on Decision and Control, 2003, **1**, pp. 25-30.
- [22] M. F. Godwin, J. K. Hedrick, R. Sengupta, "A Distributed System for the Collaboration and Control of Unmanned Aerial Vehicles", Submitted to IEEE International Conference on Robotics and Automation, 2005.
- [23] J. Finke, K.M. Passino and A. Sparks, "Cooperative Control via Task Load Balancing for Networked Uninhabited Autonomous Vehicles", Proceedings of the 42nd IEEE Conference on Decision and Control, 2003, **1**, pp. 31-36.
- [24] G. Ribichini and E. Frazzoli, "Energy-Efficient Coordination of Multiple-Aircraft Systems", Proceedings of the 42nd IEEE Conference on Decision and Control, 2003, **1**, pp. 1035-1040.
- [25] S. Spry and J.K. Hedrick, "Formation Control Using Generalized Coordinates", Accepted for Publication, IEEE Conference on Decision and Control, CDC 2004.
- [26] M. Tillerson, L. Breger and J.P. How (2003), "Distributed Coordination and Control of Formation Flying Spacecraft", Proceedings of the American Control Conference, June 4-6, **2**, 1740 – 1745.