Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Architecture and Application Abstractions for Multi-Agent Collaboration Projects

Derek Caveney* and Raja Sengupta

Abstract— **In this paper, we propose that all applications for multi-agent collaborative control, and in particular fixed-wing unmanned aerial vehicles, can be be described through three basic behaviors. They are Travelling, Watching, and Tracking. These three behaviors are constructed from two primitive actions, going somewhere and holding at a particular location. The semantics of the behaviors are clearly defined and justify why these three are chosen. The ability of these behaviors to easily program different missions through hybrid stateflow is illustrated through examples of previously-demonstrated applications.**

Furthermore, this paper outlines three levels of abstraction for a software architecture for collaborative control. In particular, we focus on an procedural-level abstraction that illustrates where the above three behaviors can be used to define applications. This level of abstraction displays software modularity such that different members of a research group can work independently while maintaining a structure that allows successful experimental implementation on the multi-agent platform. Moreover, this modularity allows project members to perform various disparate applications with the platform without considerable software modification. It also promotes software reuse after members leave the group. While focusing primarily on fixed-wing aerial vehicles, the approach presented here is equally applicable to ground and rotorcraft vehicles.

INTRODUCTION

When a reader picks up any current control journal or control conference proceedings, they are overwhelmed by the amount of attention presently devoted to the topic of collaborative control between multiple agents. Many interesting results concerning task allocation, path planning, obstacle avoidance, trajectory following, and team cooperation are presented. Furthermore, novel architectures to implement these results on experimental platforms have been proposed by various research groups ([5],[4]). The growing trend is to eliminate any centralized decision behavior, and instead distribute task allocation, path planning, and other autonomous responsibilities directly to agents, thereby making them ”intelligent”.

Regardless of where the decisions take place, most architectures are specific to the proposed application, while maintaining a general hierarchy similar to that of Fig. 1. We also don’t stray far from this basic information flow. However, unlike previous work where most layers are left as vague descriptions or ignored completely, while one or two is the focus of the publication, we attempt to give structure to all layers. In particular, we introduce structure to the two fundamental layers for collaborative control of autonomous agents, namely Collaboration and Trajectory Generation.

The overall structure, in its modularity and abstract nature, allows for project members to focus on one part of the large problem, while satisfying integration requirements of the entire system. This permits flexibility in the experimental platform and portability of work during changes in personnel. The argument presented in this paper is that applications of collaborative control can be abstracted to a level that can be described by two primitive motions and, subsequently, only three distinct, strictly-defined, behaviors. Collaboration dictates when agents transition between these three basic behaviors. We argue that all collaborative applications can be defined through hybrid automata built from these basic behaviors and present three example applications previously investigated at the University of California, Berkeley to support our hypothesis.

In our own previous experience, members of the Collaborative Control of Unmanned Vehicles (C3UV) at the University of California, Berkeley, would develop processes that ran on the PC-104 based payload once the unmanned aerial vehicle (UAV) was in flight. Successful real-time demonstrations of convoy protection [7], obstacle avoidance [3], and road following [2] have been performed. However, these processes would be written by the project members such that they oversaw the entire aircraft behavior during the particular application. Specifically, all layers of Fig. 1 would be addressed by the process written for the application. Additionally, collaboration between vehicles was programmed prior to the flight with few decisions made in the air.

After a publication had been made, or the member(s) left the group, the process and its code could quickly become unsupported by the research team. If a similar application appeared in the future, the process would have to be rewritten in such a way that the (new) member(s) responsible knew the functionality of the process. In an attempt to get things flying and demonstrated, abstractions of the problem to higher levels was omitted. In this paper, with past experience as our guide, we now return to the problem of abstracting the collaborative control of multiple agents to the fundamental elements. By adding simple structure to the problem, we succeed to modularize the elements that members can pursue in parallel.

The argument begins with abstracting the primitive actions

* Corresponding Author

D. Caveney is a Postdoctoral Researcher with the Center for Collaborative Control of Unmanned Vehicles, University of California, Berkeley, Berkeley CA 94720 caveney@vehicle.me.berkeley.edu

R. Sengupta is a Professor with the Center for Collaborative Control of Unmanned Vehicles, University of California, Berkeley, Berkeley CA 94720 sengupta@ce.berkeley.edu

This work is supported by the ONR grant #N00014-03-C-0187, SPO #016671-004

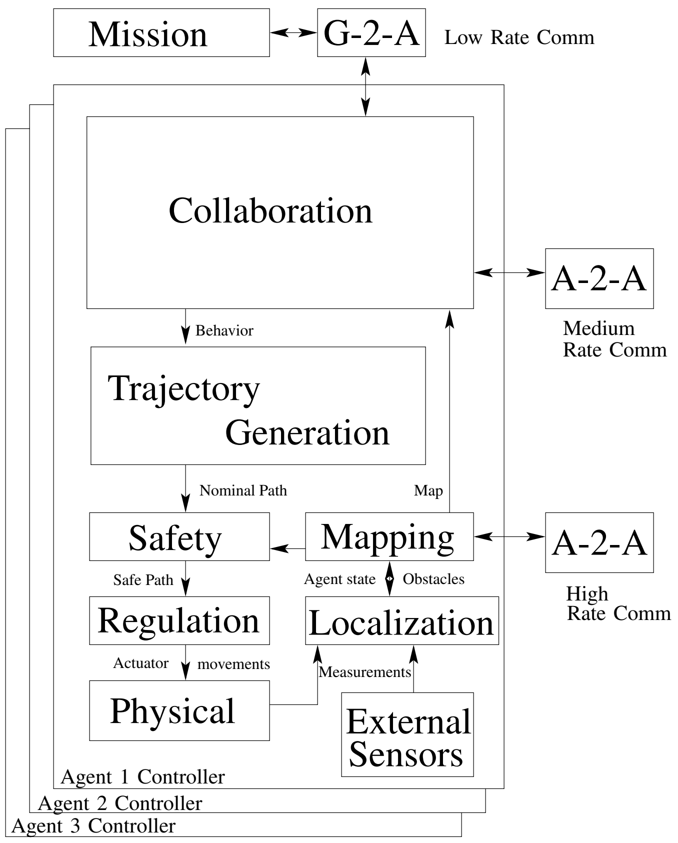

Fig. 1. Block diagram of the general layer abstraction for the motion management architecture of an unmanned vehicle

of unmanned vehicles to one of two characters, going somewhere or holding at a current location. These two primitives can be combined into countless applications, but we argue that there are only three basic applications the unmanned vehicle will perform. These are Travelling, Watching, and Tracking behaviors. Travelling is visiting a finite set of points or traversing an open path ONCE. Watching is visiting a finite set of points or traversing a closed path REPEATEDLY. Tracking is following a particular single point (static or moving) INDEFINITELY. The carefully chosen semantics of these three behaviors allow for complicated applications to be programmed through stateflow diagrams. The behaviors define the appropriate primitives through specifying the dimension and boundary conditions of trajectories. This connection can be programmed into a library of trajectory generating functions. Collaboration enters the picture by specifying the transitions within the stateflow diagram of individual agents. Moreover, collaboration enters into the safety layer when agents share their own state estimates and local obstacle maps.

The inspiration behind this work stems from the paper by Varaiya [8], where he characterizes the behavior of vehicles on an autonomous highway by simple lane-change primitives and shows how a control architecture can be built up from them. In Table I, we produce a table for our proposed unmanned vehicle architecture that is similar to one found in Varaiya's work. Parallelling his hierarchial architecture, we see that as we go up our layered architecture in Figure 1, the frequency of decisions decreases, more abstracted information is used, and the impact of decisions affects greater numbers of agents.

In the following sections, we try our best to illuminate the reader of the abstractions introduced above. In Section I, we introduce three levels of software abstraction necessary for agent-based collaboration. We particularly emphasize the modularity of the lowest abstraction that allows different project members to work on different layers simultaneously. Some members may work with knowledge of the final application, and some may work without. Regardless, their modules integrate successfully at the system level. In Section II, the semantics of our abstraction of all collaborative applications to three basic behaviors are clearly defined. The differences in these behaviors are compared and the reasons behind their inclusion in our abstraction are justified. Subsequently, we present three examples of these behaviors defining multi-agent collaborative applications in Section III. Conclusions and future work follow in the final section, Section IV.

I. THREE LEVELS OF ABSTRACTION

At the highest level of abstraction, a software architecture for one autonomous vehicle can be broken into the three areas: guidance, navigation, and control. The area of guidance concerns itself with producing appropriate trajectories for the vehicle that reflect knowledge of the terrain, the vehicle's capabilities, and collaboration between different vehicles. Control takes these trajectories and modifies them to ensure safe, collision-free, motion. Subsequently, it regulates the vehicle to follow the safe route. Navigation builds knowledge of the vehicle's own state and the surrounding environment, and the vehicle's position within this environment. The knowledge can originate from on-board sensors or communicated information from other agents. At the root of this abstraction is a supervisor who determines the way guidance should be performed for the agents assigned to a particular application. This abstraction is best shown in a feedback block-diagram, as in Fig. 2.

The second level of abstraction is that shown in Fig. 1. Its resolution is to that of layers. Collaboration and Trajectory Generation fall under Guidance. Safety and Regulation are issues dealt under Control, while the Localization and Mapping layers undertake Navigation duties. The Physical layer connects Control and Navigation through the interplay of actuators and sensors.

Overseeing the transfer of information between agents are communication protocols. These modules are stand-alone for a small group of members to investigate without concern of the final applications. As long as the essential information is decided upon by the research group as a whole, the protocols for the varying rates of communicated data can be developed in parallel with the other layers. In the UAV scenario, there are two types and three rates of communication. The first

TABLE I
DIFFERENCE AMONG THE LAYERS

Layer(s)	Time Scale	Information Span and Spacial Impact
Mission	Once every 15 minutes	Fleet-wide behavior
Collaboration and Traj. Gen.	Once every minute	Group-wide for assigned task
Safety and Mapping	Once every second	Local neighborhood of agents
Regulation and Localization	Fraction of a second / vehicle time constant	Individual agent

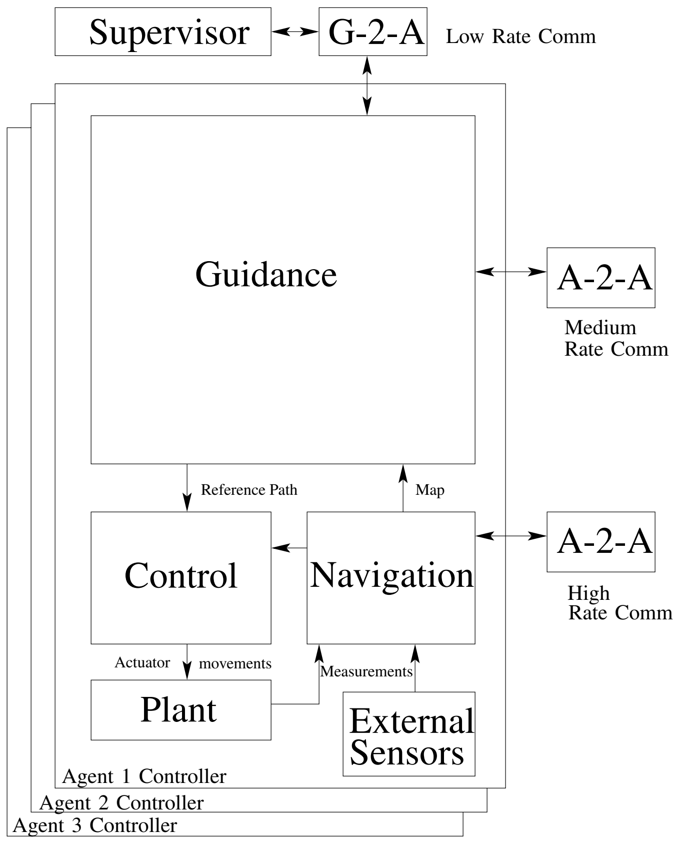

Fig. 2. Block diagram of the highest level of abstraction for the motion management architecture of an unmanned vehicle

type is Ground-to-Air (G-2-A) communication and is performed at the lowest frequency. G-2-A is the communication link between humans and their fleet of unmanned vehicles. It transmits the desired missions to members of the fleet, while not necessarily allocated individual tasks for the different members. For example, a G-2-A communication might be to patrol a 5-mile stretch of highway. No actual assignments has been given to particular vehicles, but the mission goal has been presented to the fleet.

The current frequency of G-2-A communications for the C3UV aircraft is 1Hz with kilobyte-size packets. Conversely, the Air-to-Air (A-2-A) communications being developed under the 802.11b protocol operate at transfer rates of 11Mbps. The A-2-A, which can also be thought of as Agent-to-Agent, communication performs two tasks at differing rates. The first is to communicate task assignment and reassignment processes and decisions between the aircraft. This happens on the order of minutes. Conversely, the second task is performed on the order of seconds. This communication transmits agent and local obstacle state information between neighboring agents for simultaneous map building by all agents and collision avoidance.

The final level of abstraction is procedural. It the implementable, but modular so that different members can work on different aspects. It is the one where we will introduce our abstraction on behaviors. What is argued in this paper is that although this level of software abstraction is of highest resolution, only a few basic functionalities can describe all complex applications encountered by unmanned vehicles. The basic procedural building blocks are shown in Fig. 3. Furthermore, as long as each building block covers its basic functionality, however simple, a single experimental platform can be used to implement multiple applications efficiently and reliably.

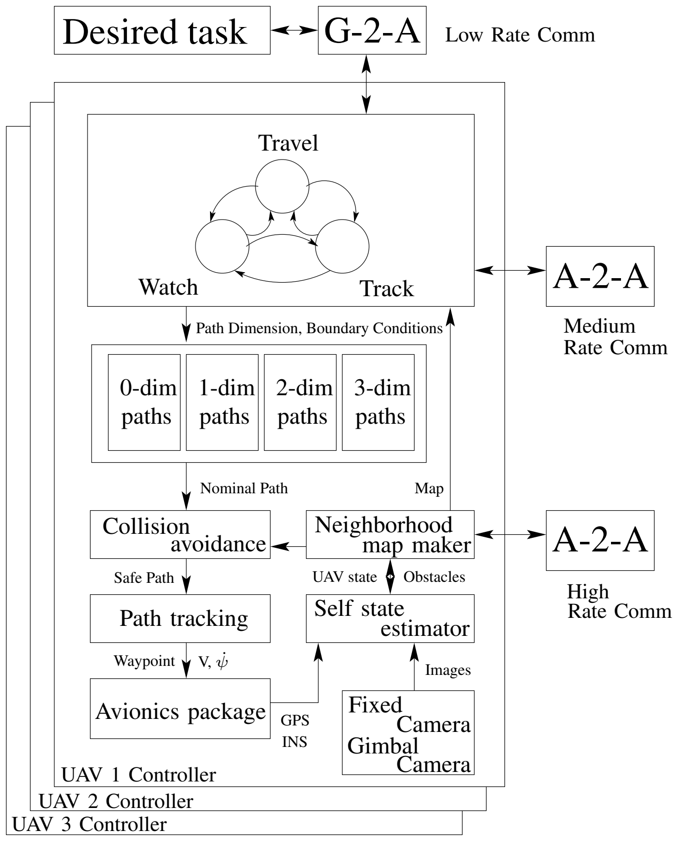

Fig. 3. Block diagram of the procedural level of abstraction of the motion management architecture of an example unmanned aerial vehicle

II. SEMANTICS

The contribution of this work appears when we look at the procedural abstraction for Guidance of unmanned vehicles. We argue that Guidance can be abstracted to three basic behaviors and these behaviors can all be built from the following primitive motions: go to a location or holding at a location. All functionality of a particular application of unmanned vehicles is simply defined by how the three basic behaviors interact and how the primitive motions are undertaken. This functionality is programmed into the Collaboration and Trajectory Generation layers. The remaining blocks of Fig. 3. are common to all applications and can be written by different members, independent of the final application. Of course, these blocks are dependent on the particular UAV platform. Furthermore, the possible applications available to the fleet of UAVs is dependent on what information the Sensory Layer can provide.

The collaboration layer is succinctly described by the general-case stateflow diagram of Fig. 3. The three types of state behavior are summarized in Table II. With respect to its interaction with the operating environment, Travelling is an open-loop behavior where the unmanned vehicle is trying to navigate through its landscape. Its objective usually focuses on time, i.e. how quickly it can complete its task. Conversely, Watching focuses on a spatial objective. Particularly, how well can I monitor the environment. At the collaboration layer, this behavior is also open-loop as it receives no feedback from the environment to adjust its path. Finally, Tracking is the one behavior that is closed-loop, with respect to its interaction with the environment. Its objective is based on how well it associates its path with some particular aspect of the environment.

The open-loop behaviors, Travelling and Watching, are with respect to their objectives and the operative environment. These do not exclude such necessary closed-loop aspects as path following and disturbance rejection, which are performed in lower layers. Furthermore, Travelling is the only behavior that can have finite duration. Both Watching and Tracking continue until something says to stop.

Each behavior can define path dimensions and the associated path parameters for the Trajectory Generation layer. The path dimension defines the qualitative type of trajectory, (0, 1, 2, or 3), whereas the path parameters are the quantitative arguments for the type of trajectory. The semantics of the various types of path dimensions are given in Table II. These types abstract the problem to the two primitive motions, go to a location and hold at a location. Holding can occur at both stationary and moving points. Going can involve performing a space-filling trajectory such that an intermediate area or volume is visited.

The transition conditions in the collaboration layer can be collaboration-based or agent-based. This is where collaboration between agents appears. An example of a collaboration-based transition could be one when two agents negotiate the distribution of two tasks where one gets to go take a image of a particular location and one gets to follow an adversary. An example of an agent-based transition would be where one agent is surveying an area, subsequently detects a target to be tracked, and then decides it would be best to track the target itself.

III. EXAMPLES

To show the effectiveness of the abstractions presented, we now introduce three collaborative UAV applications that have been investigated at Berkeley's C3UV and how they fall within the architecture. All layers below Trajectory Generation in Figure 1 are common to the applications and programmed in `C++` by individual members, independent of the applications presented here. The integrated avionics package is a Piccolo by Cloudcap Technologies (`http://www.cloudcaptech.com`). The Loiter state is a particular instance of Track behavior when the agent loiters at a stationary point while awaiting task assignments. It is also required during launch and land procedures.

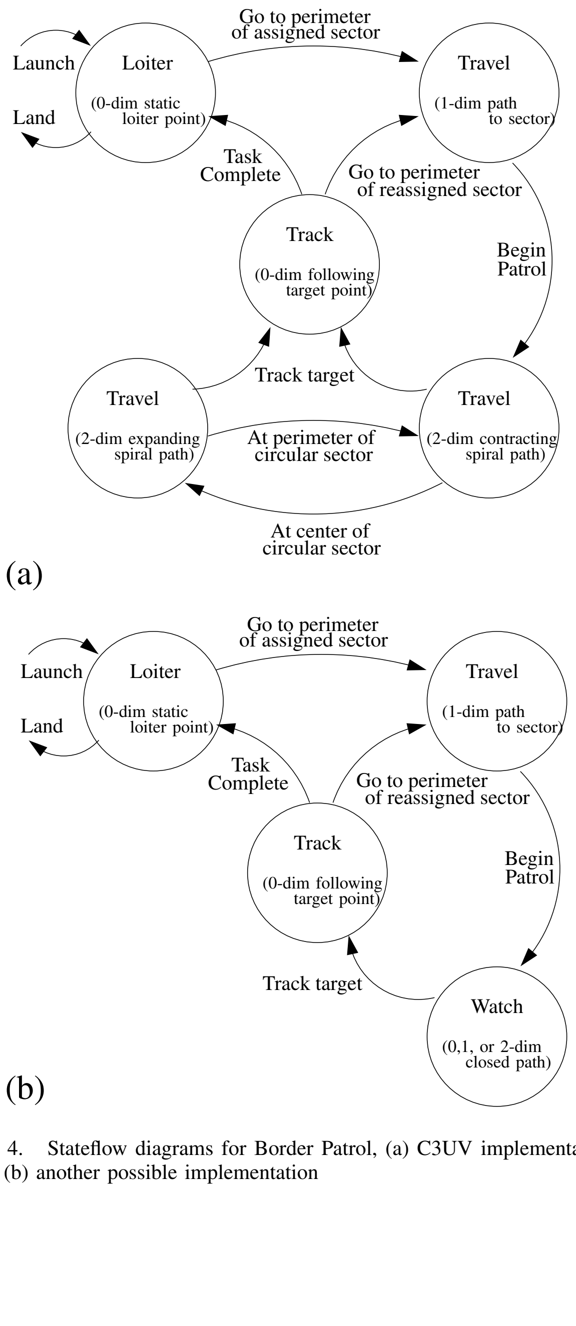
Go to perimeter of assigned sector
Launch
Land
Loiter (0-dim static loiter point)
Travel (1-dim path to sector)
Task Complete
Go to perimeter of reassigned sector
Track (0-dim following target point)
Begin Patrol
Track target
Travel (2-dim expanding spiral path)
At perimeter of circular sector
Travel (2-dim contracting spiral path)
At center of circular sector
(a)
Go to perimeter of assigned sector
Launch
Land
Loiter (0-dim static loiter point)
Travel (1-dim path to sector)
Task Complete
Go to perimeter of reassigned sector
Track (0-dim following target point)
Begin Patrol
Track target
Watch (0,1, or 2-dim closed path)
(b)

Fig. 4. Stateflow diagrams for Border Patrol, (a) C3UV implementation, and (b) another possible implementation

TABLE II

SEMANTICS OF THE BASIC BEHAVIORS AND PATH DIMENSIONS

Keyword	Definition	Optimization Focus	Interaction with Operating Environment
Travel	Cover a given set of points or open path **ONCE**	Time	Open-loop
Watch	Cover a given set of points or closed path **REPEATEDLY**	Spacial	Open-loop
Track	Stay at or with a particular point **INDEFINITELY**	Association	Closed-loop
Path Dimension	Defines type of trajectory (Qualitative)		
Boundary Conditions	Defines arguments for the trajectory (Quantitative)		
0-dim path	**Hold** at a point or follow a point		
1-dim path	**Go** between two endpoints		
2-dim path	**Go** along path that covers an intermediate area		
3-dim path	**Go** along path that covers an intermediate volume		

A. Border Patrol

Border patrol applications refer to missions where a group of UAVs is assigned to watch over a geographical region. In general, these regions are assumed long and thin, and can represent such areas as military combat fronts, security area perimeters, or national borders. The mission goal is to provide continual coverage of the assigned region by having the UAVs distribute the area amongst themselves. Coverage is typically defined in the computer vision sense. Collaboration appears both in the transitions between states and the initial distribution of the region into sectors for the individual agents to monitor. If an object of interest is discovered in one of these sectors, a single or multiple UAVs may be assigned to track the target and the total region under surveillance must be reassigned to the remaining agents. Initial efforts by the C3UV are well summarized in [4] and this implementation of border patrol is shown in Fig. 4(a). It is equally logical to divide border patrol into all three behaviors, Travel, Watch, and Track. The agent will Travel to its assigned sector. Watch can define 1-dim or 2-dim trajectories depending on the size of the sensor footprint and sector dimensions. Tracking will following a single point, which could be determined through vision feedback, attached to the target of interest. This stateflow is summarized in Fig. 4(b).

B. Convoy Protection

Convoy protection involves groups of UAVs preceding convoys of friendly, perhaps autonomous, vehicles and providing preview of the upcoming environment. Again, coverage by small UAVs is provided by computer vision. Collaborative control methodologies proposed by the C3UV are found in [7]. Again, there are various sectors around the convoy that the UAV's will negotiate to distribute amongst themselves and reallocations when agents require maintenance (ex. fuel or repair). Apart from loitering, the application only contains the Track state, where the moving point to follow is the friendly convoy's position. As the stateflow diagram Fig. 5 shows, convoy protection and the road following application addressed in [2] are similar hybrid systems where the latter follows the continuous open path of the road through vision feedback and linear structure identification.

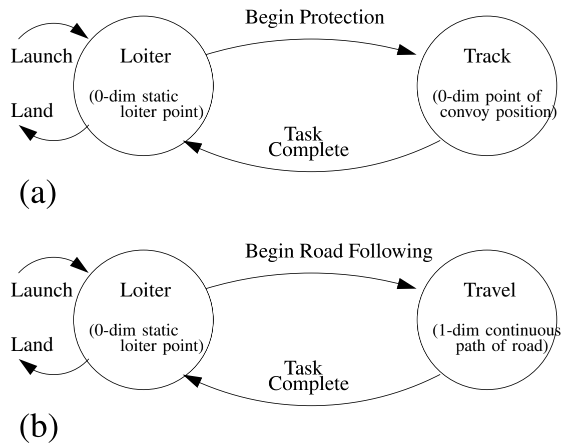
Begin Protection
Launch
Land
Loiter
(0-dim static loiter point)
Track
(0-dim point of convoy position)
Task Complete
(a)
Begin Road Following
Launch
Land
Loiter
(0-dim static loiter point)
Travel
(1-dim continuous path of road)
Task Complete
(b)

Fig. 5. Stateflow diagrams for (a) Convoy Protection, and (b) Road Following applications

C. Search and Rescue

UAV-assisted search and rescue is a topic investigated in [6]. It involves multiple fixed-wing UAVs following a piloted U.S. Coast Guard helicopter to increase the combined coverage and speed through the search area. The helicopter is performing the standard expanding square search pattern [1]. Contrary to searching with a group of UAV's alone, this search and rescue application does not just contain Travel behavior. Actually, in the semantics defined in this work, UAVs only switch to Travel behavior when the helicopter pilot chooses to turn a corner. At corners, the UAV's must leave the side of the helicopter and follows one of four predefined cornering paths. At all other times, each UAV tracks a moving point that parallels the helicopter's position. This state structure is summarized in Fig. 6.

IV. CONCLUSIONS AND FUTURE WORK

The argument presented in this paper is that collaborative control of autonomous agents can be abstracted to a level that can be described by two primitive motions and, subsequently, only three distinct, strictly-defined, behaviors. The two motions are going to a location or holding at a location and they can be used to define Travelling, Watching, or Tracking behaviors. We argue that many collaborative applications can be defined through these basic building blocks and present three example applications previously investigated at the University of California, Berkeley.

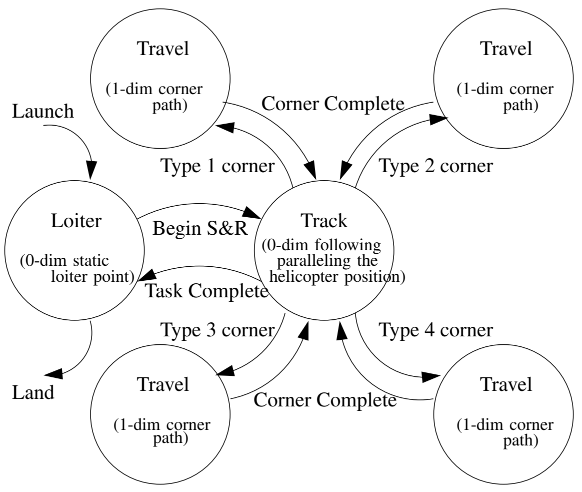

Fig. 6. Stateflow diagrams for a UAV-assisted Search and Rescue application [6]

This work proposes an architecture which is applicable to multiple agents performing heterogenous tasks. One question that arises here is that formation (flight) control and stability has received considerable attention lately in academia and industry alike, and how or where does formation control fit into the architecture. Indeed, one might argue that an additional layer is needed between the Mission and Collaboration layers. We disagree. Instead, we argue that a formation can be treated by a single agent and its Safety and Regulation layers are responsible for all formation aspects. For instance, when a formation communicates its location to other agents, it would also transmit its 3-dimensional size for collision avoidance purposes.

Currently, Berkeley's C3UV is implementing this architecture for a set of demonstrations to take place in the Summer of 2005. The stateflow logic will be programmed using `Teja`, which is suitable for real-time hybrid systems. Correctly identifying and standardizing the information structure that is passed between layers over all programmers is paramount. The authors are presently producing a coding standard manual for all students to which all programming must adhere. Failure to follow the guidelines predicates whether a certain student's code will run on the experimental platform.

References

[1] *U.S. Coast Guard Appendum to the United States National SAR Supplement (CGADD).* COMDTINST. M16130.2C, Chapter 3 and Appendix H, http://www.uscg.mil/hq/g-o/g-opr/manuals.htm, 1999.

[2] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta. Vision-based road-following using a small autonomous aircraft. In *IEEE Aerospace Conference*, Big Sky, MT, March 2004.

[3] E. Frew, S. Spry, T. McGee, X. Xiao, J.K. Hedrick, and R. Sengupta. Flight demonstrations of self-directed collaborative navigation of small unmanned aircraft. In *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop, and Exhibit*, Chicago, IL, September 2004.

[4] A. Girard, A.S. Howell, and J.K. Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. In *IEEE Conference on Decision and Control*, Nassau, Bahamas, December 2004.

[5] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How. Coordination and control experiments on a multi-vehicle testbed. In *American Control Conference*, Boston, MA, June 2004.

[6] A. Ryan and J.K. Hedrick. A mode-switching path planner for uav-assisted search and rescue. In *Submitted to IEEE Conference on Decision and Control*, Seville, Spain, December 2005.

[7] S. Spry, A. Girard, and J.K. Hedrick. Convoy protection using multiple unmanned air vehicles: Organization and coordination. In *Accepted to American Control Conference*, Portland, OR, June 2005.

[8] P. Varaiya. Smart cars on smart roads: Problems in control. *IEEE Transactions on Automatic Control*, 38(2):195–207, February 1993.