Proceedings of the 2001 Performance Metrics for Intelligent Systems (PerMIS) Workshop, in association with IEEE CCA and ISIC, Mexico City, Mexico, September 4, 2001.

Hierarchical Architecture for Coordinating Ground Vehicles in Unstructured Environments

Alberto Lacaze
Intelligent Systems Division
National Institute of Standards and Technology
lacaze@cme.nist.gov

Abstract— **This article presents a hierarchy of planners that can be used to coordinate multiple autonomous vehicles for different applications. The particular architecture reduces complexity and creates a constrained representation that in turn generates a wide variety of complex behaviors. This article will concentrate on the upper levels of the hierarchy assuming that the autonomous mobility tasks can be executed by the lower levels of the hierarchy. A particular set of examples for the US Army's Demo III project will be presented.**

Keywords— **Planning, emerging behaviors, complexity, multiresolutional hierarchical control, Real-time Control Systems (RCS).**

I. Introduction

THE problem of coordinating multiple autonomous platforms has been thoroughly studied in the literature from operations research to artificial intelligence.

The manufacturing and operations research literature shows a long history of coordinating multiple manufacturing cells to optimize factory production [1]. Most of these methods are manufacturing domain dependent and do not always easily transfer to mobile vehicles in unstructured environments.

Another field of research that has historically created coordination of mobile vehicles can be found for aerial platforms. Methods for closed coupled formations of aerial vehicles to minimize drag have been studied using linear dynamic models [2], [3], and using non-linear models [4]. These approaches rely heavily on the dynamics of the airplanes to create classical control feedback techniques that theoretically warrant the stability of the formations.

Some attempts at controlling multiple vehicles have been in structured environments using behavior-based approaches.

The predecessor to the current system employs a behavior based approach was implemented for the Demo II project [5]. Since then, the behavior based approach has been abandoned and a hierarchical architecture based on Real-time Control Systems (RCS) is currently in use for the Demo III program. The main objective of the Demo III program is to create an autonomous scouting vehicle capable of traversing an unstructured off-road environment. Behavior-based systems like [5], [6], [7], [8] share many common components with hierarchical architectures with some important differences. They integrate several goal-oriented behaviors simultaneously. In most cases several behaviors are generated, and an arbiter or decision maker weighs these behaviors to create an "intermediate" behavior that better matches the cost criteria. The advantages of these systems is that they create interesting group behaviors in simple environments where the coordination can be done relying on local criteria and therefore require simple world representations. Some examples of flocking and schooling behaviors are presented in [9]. The downfall of these architectures is that, when applied to complex environments, the implementation of each of the many possible behaviors becomes cumbersome and situation dependent; and the arbiter rapidly increases in complexity.

On the other hand, hierarchical systems create a more explicit world representation. The cost criteria are used to evaluate a model of the system traversing the predicted world representation. In most cases only one behavior is generated at each level. First, a very coarse behavior is generated, and then this same behavior is refined at each level of the hierarchy. Proponents of hierarchical architectures argue that applying cost evaluation criteria is much easier to resolve using a complete representation as opposed to dealing with multiple, sometimes contradicting, sets of behaviors. However, complex world representation and the complexity of testing plan combinations make the implementation of hierarchical systems challenging. Both architectures contain reactive and deliberative (planning) components. Hierarchical architectures tend to lean towards planning solutions because they have a representation that allows the prediction components

A. Lacaze is with the Intelligent Systems Division of the National Institute for Standards and Technology. This work was partly funded under Cooperative Agreement No. 70NANB7H0020 with the University of Maryland, College Park

Fig. 1. Two Demo III vehicles performing an autonomous mission

necessary for planning. Behavioral approaches tend to be more reactive in nature, which is sufficient in simple environments.

Other approaches taken in the literature include using classical control and stability techniques. However, since most mobile vehicles are non-holonomic, they cannot be asymptotically stabilized by smooth static-state feedback control laws. Some approaches have been taken using Lyapunov's second method [10] and smooth time-varying feedback control laws [11].

Many approaches in the literature concentrate in "tight" formations for ground vehicles where the exact location of each vehicle within the formation (parade like) is seldom used in military situations. The exact locations within the formation are very loosely followed in real scenarios and in general are not nearly as important as the overall sensor coverage, risk evaluation and distribution [12], [13], [14], [15].

Many applications assume that formations have a leader. Vehicles in the formation are then steered to a particular offset with respect to this leader [16], [17]. Other approaches allow for great freedom for the individual platforms, and the coordination is done for collision avoidance [18]. Some of these algorithms are based on the search of the configuration space [19]. [20] presents a comprehensive survey of robot coordination methods mainly concentrated for manipulators.

In this paper we will present a hierarchical system approach to controlling groups of vehicles. By imposing different constraints in the graph representation, complex militarily valuable behavior will emerge. Figure 1 shows the Demo III autonomous platforms being tested in Fort Knox in October 2000 running a hierarchical planning architecture. Figure 2 shows one autonomous platform traversing a challenging environment.

Fig. 2. Autonomous vehicle followed by manned safety HMMWV

II. Hierarchical Architectures

Hierarchical architectures based on RCS [21] make use of multiple levels of coarseness or resolution to minimize complexity. First, very coarse plans are created that look far into the future and in space. In most realistic scenarios, plans that look further into the future can only be done coarsely, because our knowledge and ability to predict outcomes rapidly deteriorates the further out we try to predict. These coarse plans are then sent to other levels of resolution where a portion of them is refined. This portion is closer in space and time to the current state, and in general, more knowledge is available. This process is continued at each level until we reach a level where very detailed knowledge, and therefore, accurate predictions can be done. These higher levels of resolution plan very detailed plans which are short in scope. Lower levels of resolution create plans and representations that include large scopes. They are coarse and there are large amounts of time to plan (and re-plan) because the representation of the world tends to change more slowly at that resolution. At higher levels of resolution, the representation and plans are much more detailed. The re-planning cycles are comparatively faster, however, the scope is small. In general, the levels of the hierarchy are designed to create a similar level of complexity for different levels. Therefore, the number of levels depend on the complexity of the problem at hand [22], [23]. For simpler systems RCS degenerates into flatter architectures similar to [24] because only a few levels of resolution are necessary to deal with the combinatorial

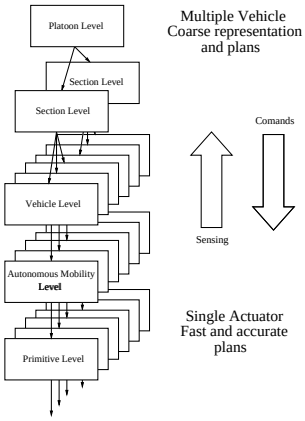

Fig. 3. RCS hierarchy for a scout platoon

complexity of the problem.

Hierarchical architectures are a very good match for military applications. Military personnel are very used to control hierarchies, and clearly understand their operation. Figure 3 shows a simple hierarchy for controlling a platoon. In the scenario presented, a platoon is composed of two sections [25]. Each section has several vehicles and each vehicle has its own vehicle level (similar to a vehicle commander), an autonomous mobility level (similar to a driver), and a primitive level which controls the vehicle. As in military structures, the commands flow from the top (platoon leader) to the bottom driver. In this paper we will concentrate on the upper two levels of this hierarchy and assume that the vehicle level and autonomous mobility levels have already been implemented in such a way that they can receive and carry out the commands passed by the upper levels.

III. Planning Algorithms

Most planning algorithms start from the following premises:
1. the universe of discourse can be subdivided into discrete states;
2. there is a starting (or current) state;
3. there are one or more goal states;
4. there is a cost associated with moving the systems from one state to another;
5. there is a cost associated with being at a state;
6. the planner must find one or more paths that will take the system from the starting state to a goal state, minimizing the cost along its motion.

Specifically, let $G = (V, E, s, f, \psi)$ be a digraph where V is a finite set of nodes, vertices or states. E has ordered pairs subsets of elements of V of called edges, that is, $E \subseteq V \times V$. $s, f \in V$, and represent a starting state and a finish state, respectively. $\psi(e)$ is a function where $e = [v_1, v_2] \in E$ and $v_1, v_2 \in V$ which computes the cost of traversing e. A planner is an algorithm $\phi(G)$ which returns a directed walk w through G (informally plan). $\phi(G) = w = (s, v_1, v_2, \ldots, v_n, f)$ where $v_1 \ldots v_n \in V$ minimizing $\sum_i \psi(e_i)$ where $e_0 = [s, v_1]$, $e_1 = [v_1, v_2]$, $\ldots$, $e_n = [v_n, f]$. $\phi(G) = \emptyset$, if there are no plans from s to f.

In most planning problems for a single ground vehicle:

- $\exists\ f : \Re \times \Re \rightarrow V$ where $\Re^2$ represents the location of the vehicle. A subsampled $\Re^2$ is used for computing the vertices of the planning graph.
- $\forall\ v_i, v_j \in V$, if $L(v_i, v_j) < thr, \exists\ e_l = [v_i, vj] \in E'$. $L(.)$ is a distance measure. In other words, vertices are connected within a vicinity.
- $E = \{e_k \in E' : Constrained(e_k) = False\}$ where $Constrained : E \rightarrow \{True, False\}$ is defined to represent the constraints that the vehicle may have (i.e. areas not allowed).

Once G is created, there are many optimal and suboptimal $\phi(G)$ described in the literature. Specifically, Dijkstra's algorithm and A* are commonly used to find these paths optimally. Both algorithms are easily implemented for replanning so that even the complexity of the second cycle is lower in the average first plan.

IV. Planning Algorithms for multiple vehicles

For two vehicles it is possible to build

$$f_4 : \Re \times \Re \times \Re \times \Re \rightarrow V \tag{1}$$

$\Re^4$ represents the position of the two vehicles. As expected, the number of elements in V and in E increases very rapidly. However, as we will see in the following examples, this is not a problem for formations. Formations create large amounts of constraints so that the number of elements of V becomes manageable.

For example, let $[x_a, y_a]$ and $[x_b, y_b]$ be two adjacent vehicle locations (i.e., $L([x_a, y_a], [x_b, y_b]) < thr$). Figure 4 shows the edges without any constraints associated with the 4D graph created for planning using the representation outlined by Definition 1.

At this stage, the number of vertices and edges that create a graph as defined could easily overwhelm the computing power, as well as the memory resources of any modern computing device. In the next few sections this paper will show how this graph is pruned by using constraints to create a graph that can be optimally searched in real time.

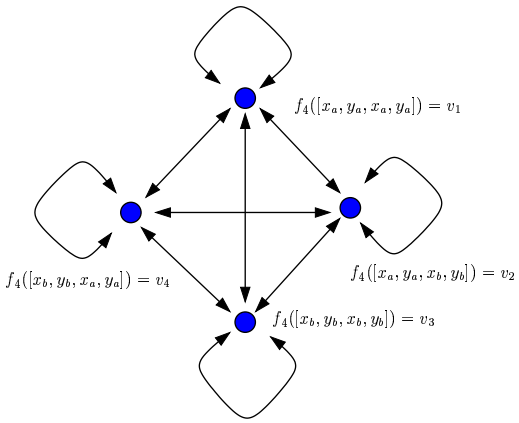

Fig. 4. Edges between two adjacent map locations for two vehicles

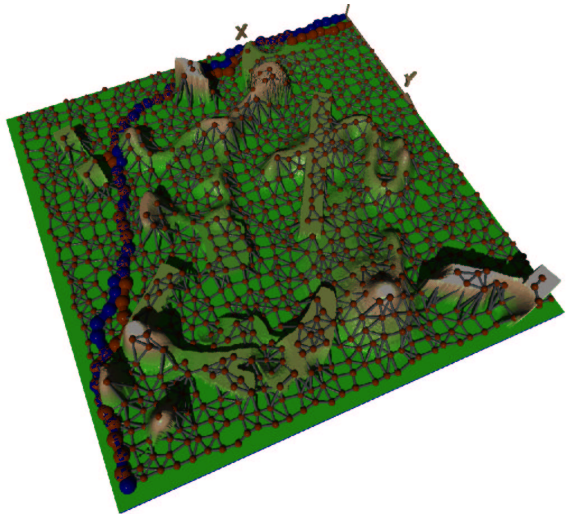

Fig. 5. Two vehicles traversing a terrain following tight distance constraints

V. Adding Constraints to Achieve Scouting Behavior

The constraints introduced in the following subsections are based on the doctrine taught to Army scouts, and it is based on [25].

It is assumed that only edges and vertices that fit the constraints are used in the creation of the graph, as opposed to creating the complete graph and then pruning it. Although similar conceptually, the amount of memory and computations required to do the former is generally orders of magnitude smaller than the later one.

A. Distance constraints

In most cases for scout maneuvering (and in most formations), there are distance constraints that must be maintained for it to be called a formation. In the case of scouting behavior, two often used constraints are as follows:

1. vehicles must not be more than l meters away from each other. A more complicated measure may actually force the vehicles in line of sight with each other. The reasons for this constraint from a military perspective are clear: cover each other, and if a vehicle gets shot, the second vehicle should find out where the shot originated.
2. vehicles must be more than m meters away from each other. This is done so that both vehicles will not be disabled by a single detonation.

Specifically, $\forall\, e_k \in E'; e_k = [[x_a, y_b, x_c, y_d], [x_e, y_f, x_g, y_h]]$:

$$\begin{aligned} &Constrained(e_k) = True \\ &iff\ L([x_a, y_b], [x_c, y_d]) > l \vee L([x_a, y_b], [x_c, y_d]) < m \vee \\ &\quad L([x_e, y_f], [x_g, y_h]) > l \vee L([x_e, y_f], [x_g, y_h]) < m \end{aligned} \tag{2}$$

Depending on the l and m chosen, this set of constraints reduces the space of search by a considerable amount. Figure 5 shows two vehicles traversing an artificially created terrain. The blue and red trails starting at the origin, represent the paths generated by the two vehicles. The underlying grid represents a two dimensional projection of the 4 dimensional graph stretch over the terrain. The map is 5 km in size, and the vehicles must be within 500 m of each other. It is possible to see from the figure that the vehicles travel mostly parallel to each other when the terrain permits, and they travel in a column when the terrain does not. There are no constraints or change in cost evaluations for the different behaviors. They travel this way because it is optimal with respect to the cost function.

Figure 6 and 7 show two vehicles traversing a artificially created maze-like and GPS generated elevation maps. The first picture shows the two vehicles with $l = 500m$ and with $l = 1500m$ (m was selected as to keep the number of nodes constant). Note that neither vehicle is following an optimal path. The paths followed by both vehicles are optimal overall. In this context, optimality refers to the fact that no other path that the two vehicles follow will give a lower cost within the given graph and constraints. This is different from

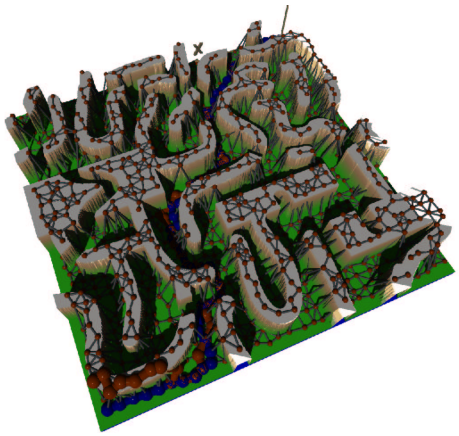

Fig. 6. Two vehicles traversing a terrain following tight distance constraints

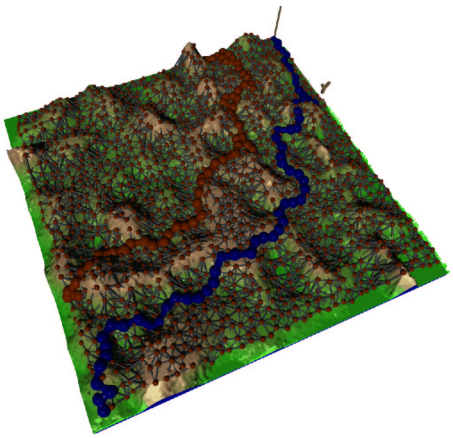

Fig. 7. Two vehicles traversing a terrain following relaxed distance constraints

the standard approach where an optimal path is found for one vehicle, and the other vehicles are constrained to the path found for the first one. In our case, the paths of both vehicles are optimized simultaneously. There are no heuristics that being used by the system (other than the constraints) that change the behavior of the system by optimizing the cost function. Very different behaviors automatically emerge depending on the terrain.

B. No Stopping Allowed

In some cases it may be necessary to only allow vehicles to stop or slow down in particular areas, and continue their moving the rest of the time. These constraints can be implemented as follows, $\forall e_k \in E'; e_k = [[x_a, y_b, x_c, y_d], [x_e, y_f, x_g, y_h]]$:

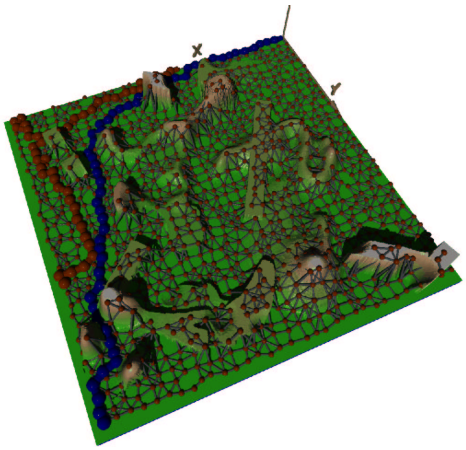

Fig. 8. Two vehicles traversing a terrain following relaxed distance constraints and a same path distance constraint

$$
\begin{aligned}
&Constrained(e_k) = True \\
&iff\ (x_a = x_e \wedge y_b = y_f) \vee (x_c = x_g \wedge y_d = y_f) \vee \\
&\quad L([x_a, y_b], [x_c, y_d]) > l \vee L([x_a, y_b], [x_c, y_d]) < m \vee \\
&\quad L([x_e, y_f], [x_g, y_h]) > l \vee L([x_e, y_f], [x_g, y_h]) < m
\end{aligned}
\tag{3}
$$

Figure 8 shows the results of applying these constraints. The starting points for one of the vehicles was modified to meet the 500m minimum distance constraint. By comparing Figure 5 to Figure 8, it is possible to see that one of the vehicles is following an optimal path while the second one is moving out of the way of the first vehicle to meet the distance constraints.

C. Leap Frog

A commonly used strategy for scouting vehicles is a "leap frog" traversal, referred to as boundary overwatch or traveling overwatch. In these cases, only one vehicle moves at a time, while the other takes an observation position over the first vehicle. If one of the vehicles is shot, the other vehicle will be paying close attention to identify the direction of the fire and other details of the encounter.

These constraints can be implemented as follows, $\forall e_k \in E'; e_k = [[x_a, y_b, x_c, y_d], [x_e, y_f, x_g, y_h]]$:

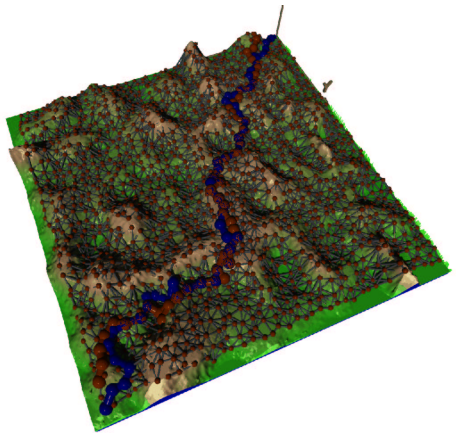

Fig. 9. Two vehicles traversing a terrain following tight distance constraints and leap frog constraints

$$
\begin{aligned}
&Constrained(e_k) = True \\
&iff\ (x_a != x_e \wedge y_b != y_f) \vee (x_c != x_g \wedge y_d = y_f) \vee \\
&\quad !LOS((x_a, y_b), (x_c, y_d)) \vee !LOS((x_e, y_f), (x_g, y_h)) \\
&\quad L([x_a, y_b], [x_c, y_d]) > l \vee L([x_a, y_b], [x_c, y_d]) < m \vee \\
&\quad L([x_e, y_f], [x_g, y_h]) > l \vee L([x_e, y_f], [x_g, y_h]) < m
\end{aligned}
\tag{4}
$$

where $LOS((x_a, y_b), (x_c, y_d)) = true$ if and only if (x_c, y_d) can be viewed (or cleared) from (x_a, y_b). LOS stands for line of sight. Figure 9 shows the results of applying these constraints. In this example, only one vehicle is allowed to move at the same time. If the cost for stopping is increased assuming that the vehicles have to take cover, the vehicles perform longer leaps. They generally stop at locations that give good visibility so that the other vehicle can perform long leaps and still be in the line of sight of the other vehicle. This explains the number eight pattern that can be seen in the path by the vehicles in the Figure 9.

VI. Coordinating Larger Numbers of Vehicles

In order to coordinate large numbers of vehicles, the dimensionality of the proposed approach becomes large. Although the number of constraints grows, this may not be enough to create small enough graphs for real time usage. In order to coordinate larger number of vehicles, following the examples given by the military organizations, we make use of hierarchical structures. Figure 10 is a schematic of the approach. At the top of this hierarchy, the platoon level creates a very coarse plan for all sections. Representation methodology, and planning strategy are the same at this level. The main differences between the levels are the coarseness of the representation as well as features of interest, cost evaluations and constraints.

In the example shown, the platoon level not only has distance constraints, but other sets of constraints do not allow sections to overlap with each other (following military doctrine). The graph is 6 dimensional where each pair of dimensions represents a rough location of each section. For the figure, the enemy is assumed to be to the left of the image, therefore, the leftmost section carries out a "leap frog" movement, while the two rightmost sections organize into more relaxed formations.

Following the lessons evolved in military doctrine, if a larger number of entities need to be coordinated, more levels would be added to the hierarchy. It is possible to describe hierarchies as sets of rules that constrain the space of search and therefore reduce complexity. This example shows that hierarchical tools designed for human entities can easily translate to artificial systems. In this example, if the paths for all six vehicles would have been searched in one level, the number of nodes and edges required to create a similar path would have overwhelmed the memory as well as the computational capabilities of the system. In general the results would not deviate from the optimally found in the 12D space.

Opponents of hierarchical systems often mention that hierarchies have a "bottleneck". In most cases these problems are caused by poor system design. Complexity of planning and representation determine the number of levels to be used for any particular system. If a level carries too much burden, then, more levels can be created to alleviate its complexity. On the down side, hierarchies create "bureaucratic" costs of communicating representations and commands between levels. In general, these added costs are negligible compared to the savings [22].

VII. Conclusions

Autonomous vehicles have been a central point of attention in recent years. The ever increasing number crunching capabilities of modern computers, as well as the recent advancement in sensor technology are paving the way for the implementation and deployment of groups of autonomous vehicles. Therefore, the need for robust formation control will become an important factor in future military applications. This paper presented a viable solution for the planning of formations of vehicles that closely resembles military organizations. It presents a departure from the behavioral approach commonly found in the literature, with

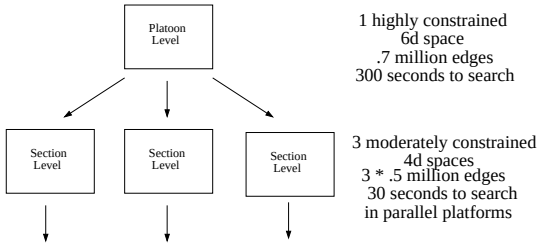

Fig. 10. Platoon Level and section levels.

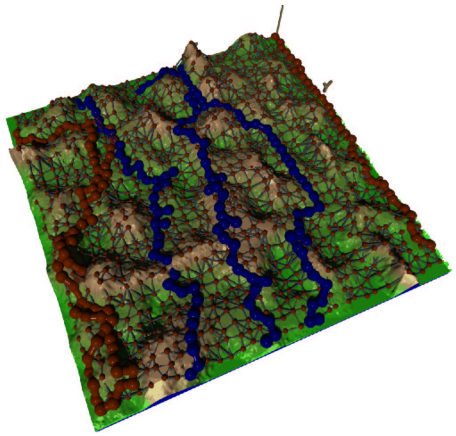

Fig. 11. A platoon formed of 3 sections with 2 vehicles each performing different section behaviors

some specific advantages:

- The system performs formation planning for multiple vehicles at the same time, as opposed to planning for one and having the others attached by control laws. In the paths created by these graph search techniques are not susceptible to the local minimums that can easily be found in ad hoc heuristics (bridges and multiple obstacles) because of their larger scope of temporal and spatial representation.
- The performance of the system is optimal within the graph representation and the constraints allocated.
- All levels shown in the examples can be easily implemented in desktop computers and allow for real-time operations at the shown resolutions. The shown examples create about 5×10^5 edges, seconds to create, plan and re-plan the graphs.
- The representation allows facilitates the generation of constraints to generate complex behavior that can result into into tactically correct behaviors.

References

[1] D. Brown, J. Marin, and Scherer W., "A survey of intelligent scheduling systems," *Intelligent Scheduling Systems*, 1995.

[2] A. Proud, M. Patcher, and J.J. D'Azzo, "Close formation control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Portland, OR, 1999, pp. 1231–1246.

[3] J.D. Wolfe, D.F. Chichka, and J.L. Speyer, "Decentralized controllers for unmanned aerial vehicle formation flight," in *AIAA Guidance, Navigation, and Control Conference*, San Diego, CA, 1996, pp. 29–31.

[4] C.J. Schumacher and R. Kumar, "Adaptive control of uavs in close-coupled formation flight," in *Proceedings of the American Control Conference*, Chicago, Illinois, June 2000.

[5] T. Balch and R. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 926–939, 1998.

[6] J. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," in *AAAI 1995 Spring Symposium*, 1995.

[7] M.J. Mataric, "Reinforcement learning in the multi-robot domain," *Autonomous Robots*, vol. 4, pp. 73–83, 1996.

[8] R. A. Brooks, *Intelligence without Reason*, MIT Press, 1991.

[9] C. Reynolds, "Flocks, herds and schools," *Computer Graphics*, vol. 21, pp. 25–34, 1987.

[10] Pomet92a, "Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift," *System Control*, vol. 18, pp. 147–58, Feb. 1992.

[11] H. Yamaguchi and JW. Burdick, "Time-varying feedback control for non honolomic mobile robots forming group formations," in *37th IEEE Conference on Decision and Control*, Tampa, FL., Dec. 1998, pp. 4156–4163.

[12] U.S. Army, *Field Manual No 7-7j*, Department of the ARMY, May 1993.

[13] K. Singh and K. Fujimura, "Map making by cooperating mobile robots," in *IEEE International Conference on Robotics and Automation*, 1993.

[14] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Second International Conference on Autonomous Agents*, 1998, pp. 47–53.

[15] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *IEEE International Conference on Robotics and Automation*, Apr. 2000, pp. 476–481.

[16] Y. Moskovitz and N. DeClaris, "Basic concepts and methods for keeping autonomous ground vehicle formations," in *International Conference on Intelligent Systems*, NIST, 1998.

[17] S. Sheikholeslam and C.A. Desoer, "Control of interconnected nonlinear dynamic systems: the platoon problem," *IEEE Transactions on Automatic Control*, pp. 806–810, 1992.

[18] T.Y. Li and J.C. Latombe, "On-line manipulator planning for two robot arms in a dynamic environment," *Journal of Robotic research*, vol. 16, pp. 144–167, 1997.

[19] T. Perez-Lozano, "Spatial planning: A configuration space approach," *IEEE Trans. Computers*, vol. 32, 1983.

[20] E. Todt, G. Raush, and R. Suarez, "Analysis and classification of multiple robot coordination methods," in *IEEE International Conference on Robotics and Automation*, San Francisco, Apr. 2000, pp. 3158–3163.

[21] J. Albus, "Outline for a theory of intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 473–509, 1991.

[22] Y. Maximov and A. Meystel, "Optimum design of multiresolutional hierarchical control systems," in *Proceedings of IEEE Int'l Symposium on Intelligent Control*, Glasgow, U.K., 1992, pp. 514–520.

[23] J. Albus, A. Meystel, and A. Lacaze, "Multiresolutional planning with minimum complexity," *Intelligent System and Semiotics*, 97.

[24] R. Arkin, *Behavior Based Robotics*, MIT Press, Cambridge, MA, 1998.

[25] U.S. Army, *Field Manual No 17-19 - Scout Platoon*, Department of the ARMY, May 1993.