Multi-Robot Team Design for Real-World Applications

Lynne E. Parker

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37831-6364 USA

Abstract

Real-world applications that are ideal for robotic solutions are very complex and challenging. Many of these applications are set in dynamic environments that require capabilities distributed in functionality, space, or time. These applications, therefore, often require teams of robots to work together cooperatively to successfully address the mission. While much research in recent years has addressed the issues of autonomous robots and multi-robot cooperation, current robotics technology is still far from achieving many of these real world applications. We believe that two primary reasons for this technology gap are that (1) previous work has not adequately addressed the issues of fault tolerance and adaptivity in multi-robot teams, and (2) existing robotics research is often geared at specific applications, and is not easily generalized to different, but related, applications. This paper addresses these issues by first describing the design issues of key importance in these real-world cooperative robotics applications — fault tolerance, reliability, adaptivity, and coherence. We then present a general architecture addressing these design issues — called ALLIANCE — that facilitates multi-robot cooperation of small- to medium-sized teams in dynamic environments, performing missions composed of loosely coupled subtasks. We illustrate an implementation of ALLIANCE in a real-world application, called Bounding Overwatch, and then discuss how this architecture addresses our key design issues.

Key words: Multi-Robot, Design, Behavior-based, Cooperative Robotics, Fault-Tolerance

1. Introduction

A key driving force in the development of mobile robotic systems is their potential for reducing the need for human presence in dangerous applications, such as the cleanup of toxic waste, nuclear power plant decommissioning, planetary exploration, search and rescue missions, security, surveillance, and reconnaissance tasks; or in repetitive types of tasks, such as automated manufacturing or industrial/household maintenance. The nature of many of these challenging work environments requires the robotic systems to work fully autonomously in achieving human-supplied goals. One approach to designing these autonomous systems is to develop a single robot that can accomplish particular goals in a given environment. However, the complexity of many environments or missions may require a mixture of robotic capabilities that is too extensive to design into a single robot. Additionally, time constraints may require the use of multiple robots working simultaneously on different aspects of the mission in order to successfully accomplish the objective. In some instances, it may actually be easier or cheaper to design cooperative teams of robots to perform some mission than it would be to use a single robot. Thus, we must build teams of possibly heterogeneous robots that can work together to accomplish a mission that no individual robot can accomplish alone.

Although a significant amount of research on mobile robotics has been accomplished in the last two decades, relatively little of this research has been transferred to use in real-world applications. We believe that two primary reasons for this are that: (1) previous research has not adequately addressed the problems inherent in real-world applications (such as robot failures and dynamic environments)

and (2) existing robotics research is often geared at specific applications, and uses concepts not easily generalized to different, but related, applications. This paper addresses these issues by first describing the design goals that must be met for real-world robotic applications. We present a general behavior-based architecture, called ALLIANCE, that was designed with these goals in mind, resulting in a fault tolerant, reliable, adaptive, and coherent mechanism for cooperative robot control. We discuss how this architecture achieves these design goals, and then illustrate the generality of ALLIANCE by presenting a proof-of-principle implementation.

The next section presents the design requirements that must be met in a successful cooperative robot approach. Section 3 briefly describes some related work in cooperative mobile robotics. Section 4 presents the ALLIANCE architecture that was designed to meet the requirements of robustness, reliability, flexibility, and coherence. Section 5 illustrates the applicability of ALLIANCE by presenting a proof-of-concept implementation for a bounding overwatch mission. Section 6 compares the *a priori* design requirements with the design of the ALLIANCE architecture. We conclude the paper in section 7 with some closing remarks on the design of cooperative robot teams.

2. Design requirements of a cooperative architecture

The difficulties in designing a cooperative team are significant. In [1], Bond and Gasser describe the basic problems the field of Distributed Artificial Intelligence must address; those aspects directly related to situated multi-robot systems include the following: How do we formulate, describe, decompose, and allocate problems among a group of intelligent agents? How do we enable agents to communicate and interact? How do we ensure that agents act coherently in their actions? How do we allow agents to recognize and reconcile conflicts?

The ALLIANCE architecture described in this paper offers one solution to the above questions. In addition to answering these questions, however, one of our primary design goals in developing a multi-robot cooperative architecture is to allow the resulting robotic teams to be robust, reliable, and flexible. The following subsections discuss these design requirements that we feel are particularly important in the design of cooperative robotic teams.

Robustness and fault tolerance

Robustness refers to the ability of a system to gracefully degrade in the presence of partial system failure; the related notion of *fault tolerance* refers to the ability of a system to detect and compensate for partial system failures. Requiring robustness and fault tolerance in a cooperative architecture emphasizes the need to build cooperative teams that minimize their vulnerability to individual robot outages — a requirement that has many implications for the design of the cooperative team.

To achieve this design requirement, one must first ensure that critical control behaviors are distributed across as many robots as possible rather than being centralized in one or a few robots. This complicates the issue of action selection among the robots, but results in a more robust multi-robot team since the failure of one robot does not jeopardize the entire mission.

Second, one must ensure that an individual robot does not rely on orders from a higher-level robot to determine the appropriate actions it should employ. Relying on one, or a few, coordinating robots makes the team much more vulnerable to individual robot failures. Instead, each robot should be able to perform some meaningful task, up to its physical limitations, even when all other robots have failed.

And third, one must ensure that robots have some means for redistributing tasks among themselves when robots fail. This characteristic of task re-allocation is essential for a team to accomplish its mission in a dynamic environment.

Reliability

Reliability refers to the dependability of a system, and whether it functions properly each time it is utilized. To properly analyze a cooperative robot architecture, one should separate the architecture itself from the robots on which the architecture is implemented. Clearly, if the architecture is implemented on robots that only function a small percentage of the time, one cannot expect dependable results from trial to trial. One measure of the reliability of the architecture is its ability to guarantee that the mission will be solved, within certain operating constraints, when applied to any given cooperative robot team. Without this characteristic, the usefulness of a control architecture is clearly much diminished.

As an example of a reliability problem exhibited in a control architecture, consider a situation in which two robots, r_1 and r_2, have two tasks, t_1 and t_2, to perform. Let us assume that their control architecture leads them to negotiate a task allocation which results in r_1 performing task t_1 and r_2 performing task t_2. Further suppose that r_1 experiences a mechanical failure that neither r_1 nor r_2 can detect. While r_1 continues to attempt to complete task t_1, robot r_2 successfully completes task t_2. However, although r_2 also has the ability to successfully complete task t_1, it does nothing further because it knows that r_1 is performing that task. Thus, the robots continue forever, never completing the mission. One would not term such a control architecture *reliable*, since a mere reallocation of the tasks would have resulted in the mission being successfully completed.

Flexibility and adaptivity

The terms *flexibility* and *adaptivity* refer to the ability of team members to modify their actions as the environment or robot team changes. Ideally, the cooperative team should be responsive to changes in individual robot skills and performance as well as dynamic environmental changes. This requirement reflects the fact that the capabilities of robots may change over time due to learning, which should enhance performance, or due to mechanical or environmental causes that may reduce or increase a robot's success at certain tasks. Team members should respond to these changes in performance by taking over tasks that are no longer being adequately performed or by relinquishing those tasks better executed by others. Each robot must decide which task it will undertake based on the actual *performance* of tasks by other robots, rather than on what other robots *say* that they are able to accomplish.

Robots must also exhibit flexibility in their action selection during the mission in response to the dynamic nature of their environment. Obviously, in real environments changes occur that cannot be attributed to the actions of any robot team member or members. Rather, outside forces not under the influence of the robot team affect the state of the environment throughout the mission. These effects may be either destructive or beneficial, leading to an increase or decrease in the workload of the robot team members. The robot team should therefore be flexible in its action selections, opportunistically adapting to environmental changes that eliminate the need for certain tasks, or activating other tasks that a new environmental state requires.

Coherence

Coherence refers to the performance of the team as a whole, and whether the actions of individual agents combine toward some unifying goal. Typically, coherence is measured along some dimension of evaluation, such as the quality of the solution or the efficiency of the solution [1]. Efficiency considerations are particularly important in teams of heterogeneous robots whose capabilities overlap, since different robots are often able to perform the same task, but with quite different performance characteristics. To obtain a highly efficient team, the control architecture should ensure that robots select tasks such that the overall mission performance is as close to optimal as possible.

A team in which agents pursue incompatible actions, or in which they duplicate each other's actions cannot be considered a highly coherent team. On the other hand, designing a coherent team does not require the elimination of all possible conflict. Rather, the agents must be provided with some

mechanism to resolve the conflicts as they arise. A simple example of conflict occurs whenever multiple robots share the same workspace; although they may have the same high-level goals, they may at times try to occupy the same position in space, thus requiring them to resolve their positioning conflict. This can usually be accomplished through a very simple protocol.

Clearly, multi-robot teams exhibiting low coherence are of limited usefulness in solving practical engineering problems. A design goal in building cooperative robot teams must therefore be to achieve high coherence.

3. Related work

Due to the large number of potential applications for cooperating robots, the amount of research in the field of cooperative mobile robotics has grown substantially in recent years. This work can be broadly categorized into two groups: swarm-type cooperation and "intentional" cooperation. The work in the current paper is of the second type. In "intentional" cooperation, a limited number of possibly heterogeneous robots work together to perform several distinct tasks. In this type of cooperative system, the robots often have to deal with an efficiency constraint that requires a more directed type of cooperation than is found in the swarm approach. Although individual robots in this approach are typically able to perform useful tasks on their own, groups of such robots are often able to accomplish missions that no individual robot can accomplish on its own. The general research issues of adaptive action selection, communication, and conflict resolution are of particular importance in these types of systems.

Two bodies of previous research are particularly applicable to this second type of cooperation. First, several researchers have directly addressed this cooperative robot problem by developing control algorithms and implementing them either on physical robots or on simulations of physical robots that make reasonable assumptions about robot capabilities. Examples of this research include the work of Lueth and Laengle [2], who study the issue of fault-tolerant behavior and error recovery in a distributed control architecture called KAMARA; Noreils [3], who proposes a three-layered control architecture that includes a planner level, a control level, and a functional level; Caloud *et al.* [4], who describe an architecture that includes a task planner, a task allocator, a motion planner, and an execution monitor; Asama *et al.* [5] who describe an architecture called ACTRESS that utilizes a negotiation framework to allow robots to recruit help when needed; Cohen *et al.* [6], who use a hierarchical division of authority to address the problem of cooperative fire-fighting; Ota *et al.* [7], who describe a mechanism for robot selection of strategies via several tactics; and Wang [8], who proposes the use of several distributed mutual exclusion algorithms that use a "sign-board" for inter-robot communication. Refer to [9] for a more comprehensive review of related work in multi-robot cooperation.

4. The ALLIANCE architecture

In this section, after describing our intended application domain, we describe a control architecture — called ALLIANCE — that was designed with the above design requirements in mind. Reasons for the design selections we made will be noted in this section, then summarized and compared to our initial design requirements in the following section.

ALLIANCE has been designed for multi-robot missions with the following characteristics: (1) mission tasks are distributed either spatially, functionally, or temporally; (2) mission tasks are loosely-coupled and independent (except that they may have fixed ordering dependencies); (3) broadcast communication is usually available (i.e., it may be noisy or fail at times); (4) robot teams are small- or medium-sized (i.e., up to tens, rather than hundreds or thousands of robots).

In addition, ALLIANCE is particularly well-suited for missions involving the following situations that we believe are true for most real-world applications of cooperative robotics: (1) dynamic environments;

(2) dynamic team composition (e.g., due to robot failures or to new robots being added to the team); (3) the team is composed of heterogeneous robots with overlapping capabilities.

ALLIANCE is a fully distributed architecture for fault tolerant, heterogeneous robot cooperation that utilizes adaptive action selection to achieve cooperative control for small- to medium-sized mobile robot teams. Under this architecture, the robots possess a variety of high-level task-achieving functions that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. The missions that the robot team can address under ALLIANCE are restricted to those missions which are composed of loosely coupled subtasks that are independent, but may have fixed ordering dependencies among them. We note, however, that even with this restriction, the proposed architecture covers a very large range of missions for which cooperative robots are useful.

In ALLIANCE, individual robots are designed using a behavior-based approach [10], in order to achieve robustness at the individual robot level. Under the behavior-based construction, a number of task-achieving behaviors are active simultaneously, each receiving sensory input and controlling some aspect of the actuator output. The lower-level behaviors, or competences, correspond to primitive survival behaviors such as obstacle avoidance, while the higher-level behaviors correspond to higher goals such as map building and exploring. The output of the lower-level behaviors can be *suppressed* or *inhibited* by the upper layers when the upper layers deem it necessary. This approach has been used successfully in a number of robotic applications, several of which are described in [11].

Extensions to this approach are necessary, however, when a robot must select among a number of competing actions — actions which cannot be pursued in parallel. Unlike typical behavior-based approaches, ALLIANCE delineates several *behavior sets* that are either active as a group or hibernating. Figure 1 shows the general architecture of ALLIANCE and illustrates three such behavior sets. The jth behavior set, a_{ij}, of a robot r_i corresponds to those levels of competence required to perform some high-level task-achieving function. When a robot activates a behavior set, we say that it has selected the task corresponding to that behavior set. Since different robots may have different ways of performing the same task, and therefore activate different behavior sets to perform that task, we define the function $h_i(a_{ij})$, for all robots r_i on the team, to refer to the task that robot r_i is working on when it activates its j-th behavior set, a_{ij}.

Because of the alternative goals that may be pursued by the robots, the robots must have some means of selecting the appropriate behavior set to activate. Thus, controlling the activation of each of these behavior sets is a *motivational behavior*. Due to conflicting goals, only one behavior set per robot can be active at any point in time. This restriction is implemented via cross-inhibition of motivational behaviors, represented by the arcs at the top of Fig 1, in which the activation of one behavior set suppresses the activation of all other behavior sets. However, other lower-level competences such as collision avoidance may be continually active regardless of the high-level goal the robot is currently pursuing. Examples of this type of continually active competence are shown in Fig 1 as layer 0, layer 1, and layer 2.

The primary mechanism for achieving adaptive action selection in this architecture is the motivational behavior. At all times during the mission, each motivational behavior receives input from a number of sources, including sensory feedback, inter-robot communication, inhibitory feedback from other active behaviors, and internal motivations called *robot impatience* and *robot acquiescence*. The output of a motivational behavior is the activation level of its corresponding behavior set, represented as a non-negative number. When this activation level exceeds a given threshold, the corresponding behavior set becomes active.

Intuitively, a motivational behavior works as follows. Robot r_i's motivation to activate any given behavior set a_{ij} is initialized to 0. Then, over time, robot r_i's motivation $m_{ij}(t)$ to activate behavior set a_{ij} increases at a "fast" rate (called $\delta\_fast_{ij}(t)$) as long as the task corresponding to that behavior set (i.e. $h_i(a_{ij})$) is not being accomplished, as determined from sensory feedback. However, the robots

Figure 1: The ALLIANCE architecture. The symbols in this figure that connect the output of each motivational behavior with the output of its corresponding behavior set (vertical lines with short horizontal bars) indicate that a motivational behavior either allows all or none of the outputs of its behavior set to pass through to the robot's actuators.

must be responsive to the actions of other robots, adapting their task selection to the activities of team members. Thus, if a robot r_i is aware that another robot r_k is working on task $h_i(a_{ij})$, then r_i is satisfied for some period of time that the task is going to be accomplished even without its own participation, and thus go on to some other applicable action. Its motivation to activate behavior set a_{ij} still increases, but at a slower rate (called $\delta\_slow_{ij}(k,t)$). This characteristic prevents robots from replicating each other's actions and thus wasting needless energy. Of course, detecting and interpreting the actions of other robots (often called *action recognition*) [12] is not a trivial problem, and often requires perceptual abilities that are not yet possible with current sensing technology. As it stands today, the sensory capabilities of even the lower animals far exceed present robotic capabilities. Thus, to enhance the robots' perceptual abilities, ALLIANCE utilizes a simple form of broadcast communication to allow robots to inform other team members of their current activities, rather than relying totally on sensory capabilities. At some pre-specified rate, each robot r_i broadcasts a statement of its current action, which other robots may listen to or ignore as they wish. No two-way conversations are employed in this architecture. Since we have designed the architecture for small- to medium-sized robot teams (i.e. tens of robots rather than hundreds or thousands), the issue of the communication cost for large numbers of robots is avoided. This broadcast form of communication is not designed for applications involving large "swarm"-type robot teams.

Each robot is designed to be somewhat impatient, however, in that a robot r_i is only willing for a certain period of time to allow the communicated messages of another robot to affect its own motivation to activate a given behavior set. Continued sensory feedback indicating that a task is not getting accomplished thus overrides the statements of another robot that it is performing that task. This characteristic allows robots to adapt to failures of other robots, causing them to ignore the activities of a robot that is not successfully completing its task.

A complementary characteristic in these robots is that of acquiescence. Just as the impatience characteristic reflects the fact that other robots may fail, the acquiescence characteristic indicates the recognition that a robot itself may fail. This feature operates as follows. As a robot r_i performs a task, its willingness to give up that task increases over time as long as the sensory feedback indicates

the task is not being accomplished. As soon as some other robot r_k indicates it has begun that same task and r_i feels it (i.e. r_i) has attempted the task for an adequate period of time, the unsuccessful robot r_i gives up its task in an attempt to find an action at which it is more productive. Additionally, even if another robot r_k has not taken over the task, robot r_i may still give up its task if it is not completed in an acceptable period of time. This allows r_i the possibility of working on another task that may prove to be more productive rather than becoming stuck performing the unproductive task forever. With this acquiescence characteristic a robot is able to adapt its actions to its own failures.

Refer to [13, 14] for more mathematical details of the ALLIANCE architecture, including the formal mathematical model of ALLIANCE, proofs of correction which guarantee that ALLIANCE will allow the robot team to accomplish its mission under certain conditions, and results of physical robot implementations of the ALLIANCE architecture for a mock toxic waste cleanup mission and a box-pushing mission.

5. An implementation of ALLIANCE

The ALLIANCE architecture has been successfully implemented in a variety of proof-of-concept applications on both physical and simulated mobile robots. The applications implemented on physical robots include a laboratory version of hazardous waste cleanup [15, 16, 17] and a cooperative box pushing demonstration [13, 17]. The applications using simulated mobile robots include a janitorial service mission and a bounding overwatch mission (reminiscent of military surveillance). The bounding overwatch mission is currently being transferred from simulation to a team of four physical robots in our laboratory. All of these missions using the ALLIANCE architecture have been well-tested. Over 50 logged physical robot runs of the hazardous waste cleanup mission and over 30 physical robot runs of the box pushing demonstration were completed to elucidate the important issues in heterogeneous robot cooperation. Many runs of each of these physical robot applications are available on videotape. The missions implemented on simulated robots encompass hundreds of runs each. In this section, we present the design and results of the implementation of the ALLIANCE architecture in the bounding overwatch mission.

The Bounding Overwatch mission requires a team of heterogeneous robots to dynamically divide themselves into two subgroups having equal distribution of robot types, and then to travel to the initial assembly points of their respective subgroups and determine a subgroup leader. Next, one team must head out for the next waypoint (i.e., they *bound*) while the other team monitors their progress and remains alert for danger (i.e., they *overwatch*). Once the first team reaches its waypoint, the roles of the teams switch, so that the first team overwatches while the second team bounds. This mission is motivated by a military surveillance scenario, in which a team of autonomous vehicles (such as tanks) must safely traverse an area thought to be occupied by enemy forces. This mission illustrates how ALLIANCE can be used for the fault tolerant execution of applications that involve many ordering dependencies among tasks.

Figure 2 shows the ALLIANCE-based control of the robots under the bounding overwatch mission. In this specific implementation, the robots have five behavior sets — *join-group*, *emerge-leader*, *follow-leader*, *lead-to-waypoint*, and *overwatch*. Heterogeneity of robots can occur when different robots specialize in different or additional aspects of the mission. For example, robots could specialize in locating safe waypoints, overwatching for danger, or carrying heavy supplies. For any tasks required by the mission that are not shown in Fig 2, additional behavior sets would be added to the robot control.

A typical run of the bounding overwatch mission is shown in Fig 3. At the beginning of the mission, the only behavior set whose sensory feedback is satisfied is the *join-group* behavior set. Since this task must be performed by all robot team members, the motivational behaviors in all the robots activate this behavior set at the beginning of the mission. This behavior set is important because it allows the team of robots to divide themselves into two equal subgroups. This division is accomplished using the

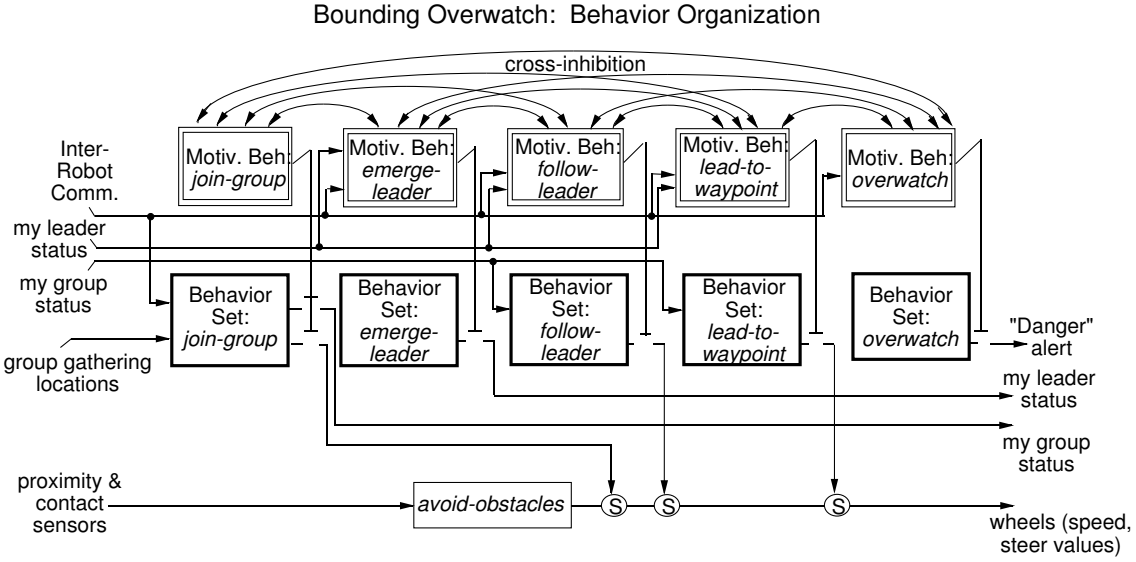

Figure 2: The ALLIANCE-based control for the bounding overwatch mission.

a simple rule in each robot. Once a robot has joined a group, it moves to the prespecified gathering location for that group. The first snapshot of Fig 3 shows the initial location of a group of eight robots — four each of two types. The second snapshot shows the robots dynamically dividing into the two groups and moving to the specified gathering locations (indicated in Fig 3 by the two small triangles closest to the robot starting locations).

The preconditions for the *emerge-leader* behavior set to activate in robot r_i are that (1) r_i has arrived at its group's initial gathering point, and (2) r_i's group does not yet have a group leader. If these conditions are met, then the *emerge-leader* behavior set is activated. The result is that the first robot to arrive at its group's gathering point becomes that group's leader. An interesting side-effect of this definition is that if, at any point in the future, robot r_i's group loses its leader, then r_i will become motivated to emerge as the team's leader. Since many other team members will also have this motivation, the relative rates of impatience across robots will determine which robot actually *does* emerge as the leader. Ideally, the rates of impatience are set such that the robots which make better leaders become motivated more quickly. A fixed priority mechanism among the robots breaks ties should they occur.

The precondition for the *lead-to-waypoint* behavior set in a leader robot is that the previous team has just bounded to its next waypoint. In order to initiate the bounding at the beginning of the mission, this condition is initially satisfied in the leader of the first team as soon as its team has collected at the initial gathering location. Thus, the leader of the first team initiates the bounding to the next waypoint. This, in turn, satisfies the preconditions of the *follow-leader* behavior set in the remaining robots on the first team. The result is that the leader robot leads the team to the next waypoint while the rest of its team follows along. The members of the second team, in the meantime, have activated their *overwatch* behavior sets, and are overwatching the first team's progress to detect any sort of danger, such as the presence of enemy agents. This scenario is shown in the third frame of Fig 3.

Once the first team's leader has arrived at its next waypoint, the completion of the *lead-to-waypoint* is broadcast. Upon hearing this, the leader of the second team's preconditions for *lead-to-waypoint* are satisfied, causing it to lead its team to its next waypoint while the first team overwatches. This continues, as shown in Fig 3, until the teams reach some prespecified destination.

The behavior-based design of the motivational behaviors also allows the robots to adapt to unexpected environmental changes which alter the sensory feedback. The need for additional tasks can suddenly occur, requiring the robots to perform additional work, or existing environmental conditions can disappear and thus relieve the robots of certain tasks. In either case, the motivations fluidly adapt to

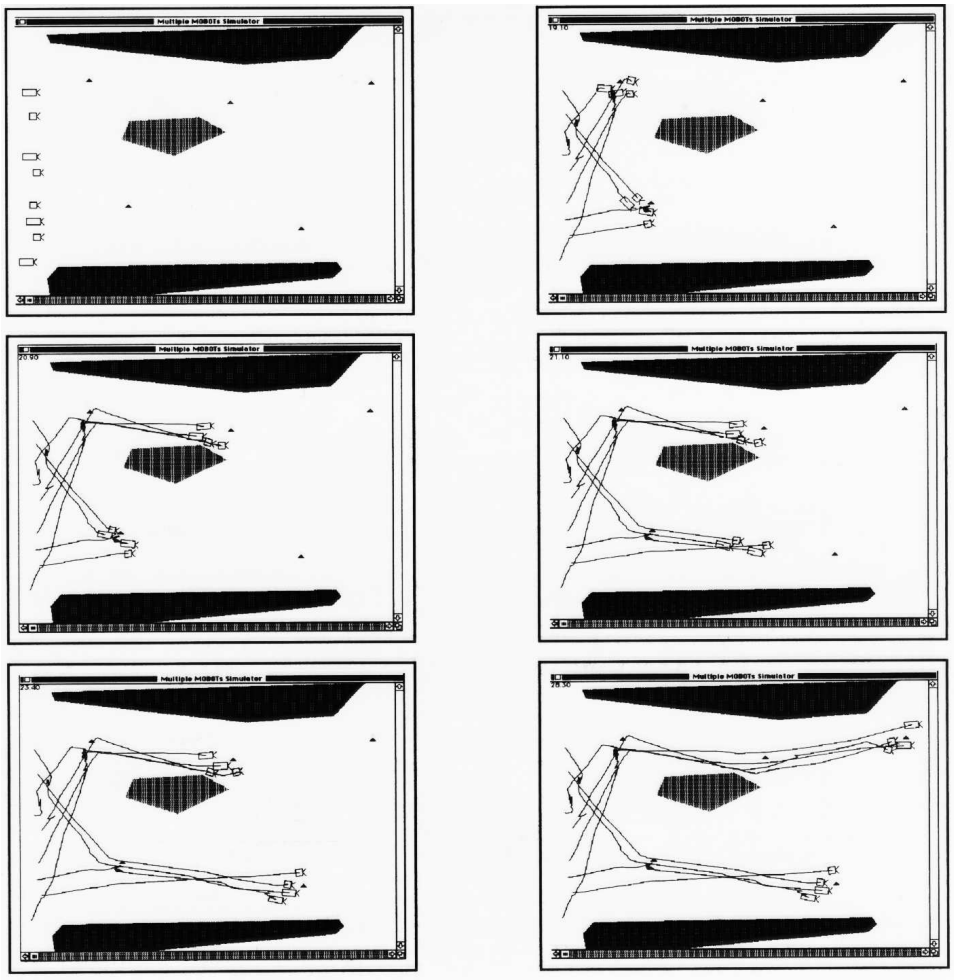

Figure 3: A typical run of the bounding overwatch mission (read the snapshots from left to right, top to bottom).

these situations, causing robots to respond appropriately to the current environmental circumstances. More complex executions of this mission have been performed in simulation that involve numerous additional robot roles (such as clearing paths, monitoring the rear of the group, searching for good waypoints, and so forth) that must be carried out by various team members. These roles can be easily and dynamically reallocated among team members with the ALLIANCE architecture when needed due to the failure of robot team members or due to increased requirements of the mission (perhaps due to an attack of enemy forces.

6. Discussion: meeting design requirements

We now review the design requirements outlined in section 2 and examine the extent to which ALLIANCE meets these design goals, using the bounding overwatch mission as an example as needed. Recall that the primary design goal was to develop a cooperative architecture that allowed heterogeneous robots to cooperate to accomplish a mission while exhibiting robustness, reliability, flexibility, and coherence. As these issues are reviewed here, note that the development of a cooperative robot architecture can actually be viewed as the development of an individual robot control architecture that facilitates a single robot's cooperation with other similarly-designed robots. Thus, we describe how each of these performance issues are addressed in ALLIANCE both from the view of an individual robot control strategy and from the view of a collective team strategy.

Robustness and fault tolerance

As described earlier, *fault tolerance* and *robustness* refer to the ability of a system to detect and gracefully compensate for partial system failures. In ALLIANCE, each individual robot is designed using a behavior-based approach which ensures that lower levels of competence continue to work even when upper levels break down. In addition, individual robots can be given multiple ways to perform certain tasks, allowing them to explore alternative approaches when met with failure.

From the viewpoint of the team, ALLIANCE first enhances robustness by being fully distributed. Unlike hierarchical architectures, since no individual robot in ALLIANCE is responsible for the control of other robots, the failure of any particular robot is not disproportionally damaging. Secondly, ALLIANCE enhances team robustness by providing mechanisms for robot team members to respond to their own failures or to failures of teammates, leading to a reallocation of tasks (such as the leader role in the bounding overwatch application) to ensure that the mission is completed. Third, ALLIANCE allows the robot team to accomplish its mission even when the communication system providing it with the awareness of team member actions breaks down. Although the team's performance in terms of time and energy may deteriorate without communication, at least the team is still able to accomplish its mission. Finally, ALLIANCE enhances team robustness by making it easy for robot team members to deal with the presence of overlapping capabilities on the team. The ease with which redundant robots can be incorporated on the team provides the human team designer the ability to utilize physical redundancy to enhance team robustness.

Reliability

Reliability refers to the dependability of a system and whether it functions properly each time it is utilized. In ALLIANCE, reliability is measured in terms of the architecture's ability to have the robot team accomplish its mission each time the mission is attempted. It can be shown that under certain conditions ALLIANCE is guaranteed to allow the robot team to complete its mission, except when robot failures eliminate required capabilities from the team — from which no architecture could recover. The presentation of this proof is beyond the scope of this paper, but more details are available in [17]. The ALLIANCE action selection mechanism thus gives a means for the robot team to achieve its mission reliably.

Flexibility and adaptivity

Flexibility and *adaptivity* refer to the ability of robots to modify their actions as the environment or robot team changes. The motivational behavior mechanism of ALLIANCE constantly monitors the sensory feedback of the tasks that can be performed by an individual agent, adapting the actions selected by that agent to the current environmental feedback and the actions of its teammates (such as the subgroup selection of robots in the bounding overwatch application). Whether the environment changes to require the robots to perform additional tasks or to eliminate the need for certain tasks, ALLIANCE allows the robots to handle the changes fluidly and flexibly. ALLIANCE enhances the adaptivity and flexibility of a robot *team* by providing mechanisms for robots to work with any other robots designed using ALLIANCE; the robots are not required to possess advance knowledge of the capabilities of the other robots.

An extension to ALLIANCE, called L-ALLIANCE [17], beyond the scope of the current paper, further extends the flexibility and adaptivity of the system by allowing robots to learn about their own abilities and the abilities of their teammates in order to improve their performance on subsequent trials of similar missions whenever familiar agents are present.

Coherence

Coherence refers to how well the actions of individual agents combine towards some unifying goal. For individual agents, ALLIANCE causes robots to work only on those tasks which the environmental feedback indicates need to be executed. Thus, ALLIANCE will not cause an individual agent to

work on some task that is not required by the mission, nor consistent with the current state of the environment.

Obtaining coherence at the team level requires that robots have some means of determining the actions of other robots and/or the effect of those actions on the environment. Without this knowledge, the robots become a collection of individuals pursuing their own goals in an environment that happens to contain other such robots. While we certainly want the robots to be able to accomplish something useful even without knowledge of other robots on the team, ideally each robot should take into account the actions of other robots in selecting their own actions.

Determining the actions of other robots can be accomplished through either passive observation or via explicit communication. Since passive action recognition is very difficult and is a major research topic in itself, ALLIANCE augments the observation skills of the robot team members through the use of one-way broadcast communication that provides each robot with an awareness of the actions of other robots, plus the ability to act on that information. With this awareness, robots do not replicate the actions of other robots, thus giving them more coherence. We note the importance of this mechanism to achieve team coherence, since when the communications mechanism is unavailable, team coherence is reduced. Refer to [12] for a more detailed discussion of this issue.

Note that a number of issues regarding the efficiency of ALLIANCE are not addressed here. Among these issues include questions of how long robots remain idle before activating a task, how to ensure that robots failing at one task go on to attempt another task they might be able to accomplish, how robots deal with having more than one way to accomplish a task, and so forth. These issues are addressed in the L-ALLIANCE extension to ALLIANCE [17] through the use of dynamic parameter update mechanisms.

7. Conclusions

In this paper, we have addressed the issue of developing robotic technologies that successfully deal with the dynamics and complexity found in real-world applications. We described the design goals of real-world robotics applications, and presented a general architecture — called ALLIANCE — that facilitates the fault-tolerant, adaptive control of small- to medium-sized teams of cooperating robots. The key characteristics of this architecture can summarized as follows:

- Fully distributed (at both the individual robot level and at the team level)
- Applicable to robot teams having any degree of heterogeneity
- Uses one-way broadcast communication; no negotiation or two-way conversations are utilized
- Recovers from failures in individual robots or in the communication system
- Allows new robots to be added to the team at any time
- Allows adaptive action selection in dynamic environments
- Eliminates replication of effort when communication is available
- Provably terminates for a large class of applications

We illustrated one implementation of this architecture by presenting the bounding overwatch mission. In this application, the architecture offers an easy way to achieve dynamic role transferral in missions involving changes in the environment or in robot team. These applications, along with the hazardous waste cleanup mission described in [14, 15] illustrate the wide variety of applications for which the ALLIANCE architecture is well-suited.

Acknowledgements

This research was funded in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-91-J-4038 at the Massachusetts Institute of Technology's Artificial Intelligence Laboratory, and in part by the Office of Engineering Research, Basic Energy Sciences (directed by Dr. Oscar Manley), of the U.S. Department of Energy, under contract No. DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp.

References

[1] Alan Bond and Less Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, 1988.

[2] T. C. Lueth and T. Laengle. Fault-tolerance and error recovery in an autonomous robot with distributed controlled components. In H. Asama, T. Fukuda, T. Arai, and I. Endo, editors, *Distributed Autonomous Robotic Systems.* Springer-Verlag, 1994.

[3] Fabrice R. Noreils. Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment. *The International Journal of Robotics Research*, 12(1):79–98, February 1993.

[4] Philippe Caloud, Wonyun Choi, Jean-Claude Latombe, Claude Le Pape, and Mark Yim. Indoor automation with many mobile robots. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 67–72, Tsuchiura, Japan, 1990.

[5] H. Asama, K. Ozaki, A. Matsumoto, Y. Ishida, and I. Endo. Development of task assignment system using communication for multiple autonomous robots. *Journal of Robotics and Mechatronics*, 4(2):122–127, 1992.

[6] Paul Cohen, Michael Greenberg, David Hart, and Adele Howe. Real-time problem solving in the Phoenix environment. COINS Technical Report 90-28, U. of Mass., Amherst, 1990.

[7] J. Ota, T. Arai, and Y. Yokogawa. Distributed strategy-making method in multiple mobile robot system. In H. Asama, T. Fukuda, T. Arai, and I. Endo, editors, *Distributed Autonomous Robotic Systems.* Springer-Verlag, 1994.

[8] Jing Wang. DRS operating primitives based on distributed mutual exclusion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1085–1090, Yokohama, Japan, 1993.

[9] Y. Uny Cao, Alex Fukunaga, Andrew Kahng, and Frank Meng. Cooperative mobile robotics: Antecedents and directions. In *Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95)*, pages 226–234, 1995.

[10] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[11] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[12] Lynne E. Parker. The effect of action recognition and robot awareness in cooperative robotic teams. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95)*, Pittsburgh, PA, August 1995.

[13] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proc. of the 1994 IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems (IROS '94)*, pages 776–783, Munich, Germany, Sept. 1994.

[14] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. Technical Report ORNL/TM-12920, Oak Ridge National Laboratory, January 1995.

[15] Lynne E. Parker. An experiment in mobile robotic cooperation. In *Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments*, Albuquerque, NM, February 1994.

[16] Lynne E. Parker. Fault tolerant multi-robot cooperation. MIT Artificial Intelligence Lab Videotape AIV-9, December 1994.

[17] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation.* PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, February 1994. MIT-AI-TR 1465 (1994).