Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Multi-Robot Communication-Sensitive Reconnaissance

Alan Wagner
College of Computing
Georgia Institute of Technology
Atlanta, USA
alan.wagner@cc.gatech.edu

Ronald Arkin
College of Computing
Georgia Institute of Technology
Atlanta, USA
arkin@cc.gatech.edu

This paper presents a method for multi-robot communication-sensitive reconnaissance. This approach utilizes collections of precompiled vector fields in parallel to coordinate a team of robots in a manner that is responsive to communication failures. Collections of vector fields are organized at the task level for reusability and generality. Different team sizes, scenarios, and task management strategies are investigated. Results indicate an acceptable reduction in communication attenuation when compared to other related methods of navigation. Online management of tasks and potential scalability are discussed.

Keywords:behavior-based robotics, internalized plan, multi-robot, reconnaisance

I. Introduction

This paper contributes a novel multi-robot method for performing communication-sensitive reconnaissance, which involves exploring an urban area in a manner that prevents a team member from becoming a lone disconnected network. This type of reconnaissance is one goal of DARPA's MARS Vision 2020 program and has implications for much of the multi-robot community. Communication sensitivity is an important consideration for teams of robots operating in dynamic and potentially hazardous environments. In particular, communicating robots may be more capable of self-rescue, better equipped to relay information back to a human operator, and have advantages in terms of localization. [3]. These types of environments require agents capable of coordinated sensing, processing, and communication [6].

Multiple architectures have been created for the purpose of autonomous navigation. Implemented systems range from purely reactive [4] to sense-plan-act [9]. Hybrid deliberative/reactive architectures attempt to address the short-comings of these two extremes [1, 5]. Other approaches include continuous calculation of a local gradient field [8] or planning only when reactive behaviors fail [12].

Payton also delineates a method for combining planning with reactive navigation [11]. In this method, a priori map knowledge becomes an enabling resource for decision-making. From his perspective, traditional plans are artificially abstracted from knowledge that often results in over- or under-specification of a mission's objectives. By minimizing symbolic abstraction, a plan for action is developed that can be used directly by a reactive agent. Payton brands this type of plan an *internalized plan*. These internalized plans differ from traditional plans by their lack of abstract symbol use and their tight representational coupling to the needs of a reactive robot. Moreover, the plans are used only as advice, where injecting world map or other types of knowledge is performed only at the discretion of the robot.

Combining plans with reactive navigation is not new. Rather than simply implementing a planning algorithm on top of a reactive architecture, the method outlined in this paper both extends and generalizes an earlier hybrid approach. In previous research [13], internalized plans were integrated with Arkin's motor schema architecture [2] using the *Missionlab* [10] behavior specification software. An internalized plan is created by running a uniform cost search algorithm to produce a gradient field on a grid mesh. The resulting vector field directs a robot from any location on a map to a goal location. This hybrid approach alleviates some of the problems associated with purely reactive systems (e.g., local minima, box canyons, and mazes) while still providing timely response to unplanned obstacle encounters. This earlier work also developed an efficient method for using sets of multiple plans in parallel, enabling a robot to focus attention on one plan over another in a given situation or via a weighted combination of plans. By stacking multiple vector fields on top of one another, advice can be either arbitrated or weighted based on a rank ordered attention mechanism (fig. 1).

Building from this initial research, new procedures and techniques for plan selection, organization, and coordination are developed that can potentially be extended to novel environments and generalized across a wide variety of domains. This approach shows promise as a method for coordinating teams of robots while performing communications-sensitive reconnaissance.

II. Method Overview

Our method operates on the premise that reconnaissance of a large urban area can be reduced into a collection of smaller reconnaissance tasks. Moreover, if each of these tasks is sensitive to communication attenuation then, overall, the entire operation will be sensitive to communication attenuation. Naturally there may be many different tasks; circle a target area, perform reconnaissance within a building, or exit a location, to name a few. Each individual task may also need to be repeated several times, differing only by some parameter, such as location. When a task is repeated most, if not all, of the

This research is funded under DARPA/DOI contract #NBCH1020012 as part of the MARS Vision 2020 program.

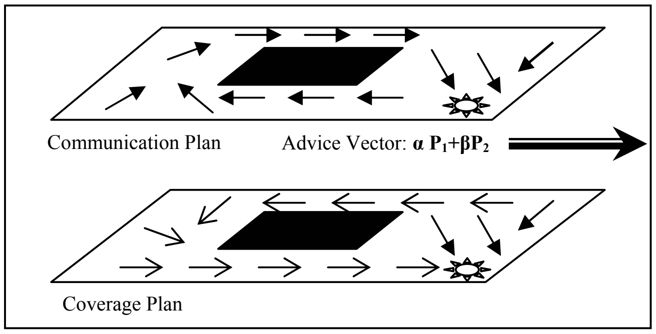


Figure 1. Plans used in parallel. The top plan represents a communications plan. The bottom plan represents a coverage plan. Output advice is determined via arbitration or weighted summation where α is the weight of plan P_1 and β is the weight of plan P_2.

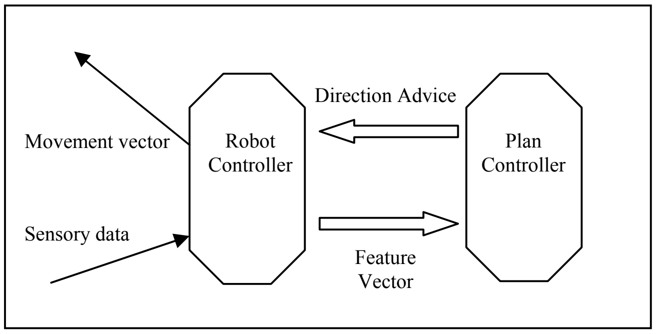


Figure 2. Interaction of the robot controller, plan controller, and feature vectors. The robot controller maintains control of the robot but receives directional advice from the plan controller. The plan controller, in turn, receives information regarding the status of the current environment encoded in a feature vector and presents directional advice to the robot controller.

underlying structure that composes the task may be similarly repeated. Hence by intelligently developing and combining simple communication-sensitive reconnaissance tasks this method can be made to perform reconnaissance on a much larger scale. For example, reconnaissance of one particular building may differ from reconnaissance of another building in the details of the rooms or the intermediate locations to be navigated, but not regarding the overall procedures performed. In this research, a task is represented by a plan element, and repetition of the same task is represented by different instances of that same element.

The preceding argument has important consequences from a planning perspective. Although the moment-to-moment nuances of the environment may change radically, the procedures, steps, and milestones for accomplishing a mission will likely not change. Granted some plans may be made untenable by changes in a dynamic environment. In this case it is often acceptable for the team to recognize that its plan is invalid and give up (and replan) rather than attempt to solve a problem for which it has little or no resources. This line of reasoning draws from work on cognizant failure [7]. In our case since planning is used only as advice to a reactive controller, the system should remain robust to dynamic and unaccounted for obstacles and impediments. Moreover, even the method by which the "advice" is realized may not be important, as long as the underlying reactive controller has the final determination of which heading and velocity to select.

Team coordination is another important aspect for multi-robot reconnaissance. This approach provides mechanisms for both between-task and within-task coordination. Between-task coordination is accomplished by restricting the transition from one task to another until all robots have completed the preceding task. Within-task coordination is accomplished using specifically designed progress points or stages for the task. Movement from one progress point to another is similarly restricted until the preconditions of a transition function are satisfied. The larger components of this method are described below and depicted graphically in figure 2.

One component is the plan controller. This component is devoted to four tasks: 1) managing the plan elements which compose the overall mission, 2) coordinating the progression from one plan element to the next 3) determining the role each robot will play when executing an element and 4) communicating the plan controller's advice to the robot controller. Pseudocode for this procedure is provided in figure 13.

As mentioned, this software represents tasks as individual plan elements. These elements can be managed in several ways. The overall mission could, for example, demand a strict ordering of tasks. Or, on the other hand, a least commitment ordering of tasks might be acceptable (see figure 13 for pseudocode). The plan controller also coordinates the transition from one plan element to the next. This guarantees the completion of one task (either successfully or unsuccessfully) prior to beginning another task. The plan controller may need to assign robots to particular roles if the underlying task demands it. Currently, the controller assumes the robot team to be homogeneous, but we foresee no difficulty extending this approach to heterogeneous teams and hope to address this task in future work. Finally the planning system's advice is communicated to the robot controller via a unit vector.

The robot controller executes the underlying reactive system, in this case *Missionlab's* motor schemas. The robot controller interfaces with the plan controller receiving its advice in the form of an egocentric unit vector. The robot controller maintains the option to ignore plan advice at any time and provides the plan controller with positional information and sensor-derived feature vectors.

Feature vectors provide a snapshot of the robot's environment to the planning apparatus. They contain the information necessary to produce an advice vector. Feature vectors are created by the robot controller based on incoming sensory information.

III. PLAN ELEMENTS

Plan elements have already been described as the individual tasks into which the overall mission is decomposed. Each plan element represents a recommended solution to a problem. A single element guides each robot of a team in a coordinated manner to accomplish a specific task. Many unique instantiations of a single element may exist. Figure 3 depicts the components of a plan element and figure 13 provides pseudocode.

All plan elements share a common interface format. This serves several purposes. First, it allows uniform handling and management of all elements. Second, it ensures that the

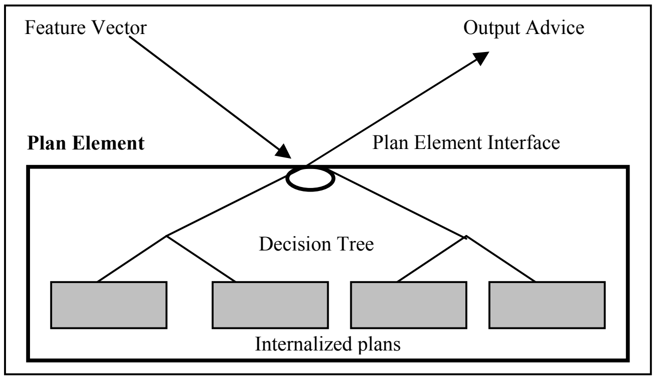


Figure 3. The internal structure of a plan element. The major components that characterize a plan element are depicted.

implementation underlying an element is restricted to the element itself. Finally, it defines and restricts the type of information that an element can produce and receive. It is expected that by using a common interface the development of additional plan elements will be made easier and standardized.

Currently the interface limits each plan element's input to a feature vector, the role of the robot, and the number of robots in the team. This allows the element to be generalized with respect to team size, the robot's role in the task, and the status of the environment. This design decision has benefits and drawbacks. On the one hand, an interface that limits input to all elements vastly simplifies the management and operation of the elements. On the other hand, because different elements may require different input, this strategy will likely fail when the number and type of elements becomes more complex. One solution might be to maintain two separate input channels: one common among all elements and one specific to each element. In any case, as the quantity of planned tasks a robot can perform increases so does the difficulty of managing and using those tasks.

The common element interface similarly limits each element's output to a unit vector with directional advice. The output is then passed from the plan controller to the robot controller. Because this method utilizes several plans in parallel additional channels for element output are not necessary. This is an important and defining characteristic of our approach. The planning system distills several rapidly changing and complex input variables into a single piece of output advice that will not require any additional processing for use by the robot controller. Moreover this is performed in real-time in a manner that is reusable from element instance to element instance.

Between each plan element's input and output, two components—a decision tree and a parallel plan—accomplish the reduction in complexity. Each element maintains a decision tree mapping the element's input to an array of gain values. Gain values are multipliers of the basic output vectors of each behavior [2], or, in this case, the relative strength of the internalized plans. When input is presented to an element by the plan controller it flows down the element's decision tree eventually resulting in an array of gains. This array will determine the influence (gain) of each individual internalized plan in a parallel-internalized plan. While the decision tree for each instance of an element is the same, the decision tree for each **type** of element may be different. In other words each

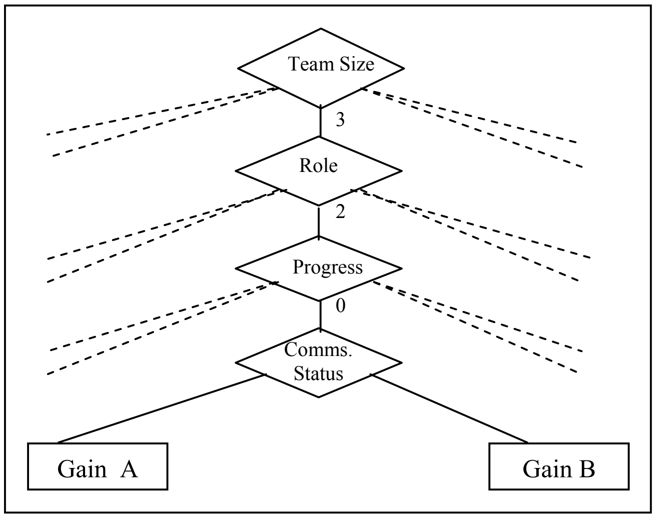


Figure 4. The decision tree used for the `ReconBuilding` element. The first node selects based on team size, the next node on role, the third nodes on progress and the final node based on communications status. Example values are displayed below and to right of each node. Gain A and B are meant to represent arbitrary gain arrays.

decision tree is task-specific but not instance-specific. The decision tree maintains the guidelines by which a task is conducted. An element's decision tree may be extremely complex or as simple as a direct pointer to a single static gain. For example, if the robot's ability to communicate is acceptable, then the decision tree may select an array of gains that favor a coverage plan over a communications plan, perhaps driving two robots in different directions. If, on the other hand, the robot's current ability to communicate is unacceptable then the resulting gain values will prefer a communication plan over a coverage plan, perhaps forcing the robots toward one another.

Figure 4 outlines the decision tree for one plan element used in this study—`ReconBuilding`. Due to space considerations only a single path through the tree is shown. The first node of the decision tree branches based on team size. Later the tree branches according to role, progress, and communication status. Arbitrary values have been included for completeness: team size-3, role-2, and progress-0. These values map to the gain arrays A and B.

The final stage in the operation of a plan element applies the array of gains determined by the decision tree to a parallel plan producing the element's output. An internalized plan [11] has already been described as a gradient field generated on a grid mesh using a map. Multiple internalized plans can be utilized in parallel by stacking individual plans on top of one another. Output advice is then determined by multiplying the advice for each individual plan at a location by a gain as determined by the decision tree. The detailed use of parallel-internalized plans appears in [13].

IV. Implementation

The system was developed with the intent of enabling teams of autonomous robots to perform coordinated communication-sensitive reconnaissance as part of DARPA's MARS Vision 2020 program. Prior to implementation it was

Figure 5. The training village located at Fort Benning GA.

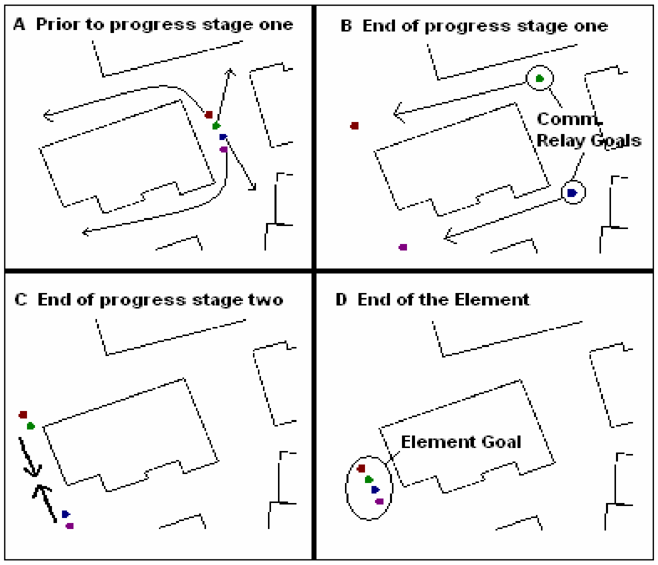
necessary to determine the types of tasks that are necessary for successful completion of a communication-sensitive reconnaissance mission. The fielded system will be tested in an outdoor mockup of a European village (fig. 5) at Ft. Benning, Georgia. A team of robots must explore this small town of less than a 0.25 x 0.25km area. Our overall role in Vision 2020 has guided the choice of tasks for the robot team. It is not maintained that these are the best, most optimal, or most characteristic tasks and associated elements for performing reconnaissance. They simply represent the tasks chosen to address this problem, and other design choices are possible.

This section is organized into three parts. First, the two major tasks are described in detail. The computational process necessary for completing each task is then explained. Finally the procedure for using a plan element is described.

A. *Description of Elements*

Two types of plan elements were defined, one for each task deemed necessary. `ReconBuilding` is an element devoted to surrounding and moving around buildings, in a communication-sensitive manner. This allows the team to explore the entire village, including alleys, streets, and passageways, with minimal lose of communication; eventually covering the entire area and succeeding in its reconnaissance mission. `MoveTo` is an element that guides the team of robots from one location to another, resorting to a contingency plan if communication fails. This element guides robots between the reconnaissance of individual buildings. Other methods, such as reactive navigation, could have been used instead. A second plan element was implemented in order to examine the generic nature of the plan element interface.

The `MoveTo` plan element directs the robots to a goal position from any location on the map. Network signal strength, along with other unused information, is input to this element by the plan controller. Its decision tree assigns a gain of 1.0 to the internalized plan leading to the goal and 0.0 to the contingency plan while the network signal strength to all teammates is above a 10% signal strength threshold level. If the network signal strength decreases below the threshold level, a gain of 0.0 is assigned to the internalized plan leading to the goal and a gain of 1.0 assigned to the contingency plan,


A Prior to progress stage one
B End of progress stage one
Comm.
Relay Goals
C End of progress stage two
D End of the Element
Element Goal
Figure 6. Progress stages for the `ReconBuilding` plan element. (A) depicts the start position and the different directions for each robot's role at this stage. (B) shows the robots in position for the transition to stage 2. (C) displays the robots in position for the tranistion to stage three. (D) depicts the robots at the goal location.

effectively switching between the plans. Once the contingency plan has been selected it continues to function within this element regardless of network signal strength. This prevents the robots from thrashing between plans at the border of a communications failure.

The `ReconBuilding` plan element is more complex. This element employs all data input from the plan controller: the robot's current position, the locations of teammates, the signal strength to each teammate, and the robot's role. The feature vector, transmitted to the element by the plan controller, contains the pertinent sensory information. The robot's role is assigned to the element when the element is instantiated. Input from the plan controller traverses the decision tree in figure 4. The first branch of the tree segregates based on team size. The team size attempting to surround the building is an important factor in determining where and when a team member should move. If the team consists of two robots the robots initially attempt to pass the object from opposite sides. Intuitively sending each robot around a different side of a building may seem improper when communication maintenance is one of the stated goals of the system. However, these tasks were designed to prefer opportunistic plan advice. Thus an attempt is made to first surround the building even if communication attenuation is likely. Other choices are possible. Three robot teams attempt to surround the building by leaving one robot behind to act as a communication relay for the other two members. Thus, in a team of three, one robot is assigned the role of communication relay, one is tasked with exploring one side of the building, and the third robot is task with exploring the opposite side. Teams of four robots, depicted in figure 6, attempt to surround the building by creating a rectangle of communicating robots around the building. Again two robots are assigned (by the plan controller) the role of exploring opposite sides of the building. The remaining two robots act as communication relays by moving to nearside positions. In this research, the number of roles a team has is equal to its team size.

Surrounding a building with a robot team requires coordination. For this reason the third node of the decision tree branches based on the robot's progress. A two robot team has no progress stages. Teams of three and four have three progress stages. The first stage begins with the robot's initial locations and ends when the robots assigned to explore opposite sides of the building have reached their assigned locations and reestablished contact with one another. The beginning of stage one for a team of four robots is depicted in figure 6a. The arrows indicate each robot's initial trajectory. Figure 6b shows the team at the end of stage one. Progress stage two guides the two-team members acting as communication relays to their final goal location. This stage ends when the communication relay robots have reached the same location as the exploring robots, depicted in box three of figure 6c. The final stage guides all of the robots the goal location.

The last node of the decision tree selects based on communication status. Acceptable versus unacceptable communication is influenced by team size, role, and progress stage. For example, in a team of four robots at the first progress stage, the exploring robot only needs to maintain a link with a particular communication relay robot. A communication relay robot, on the other hand, must maintain a link with both an exploring robot and the other communication relay robot.

The `ReconBuilding` element's parallel plan consists of four individual internalized plans: a contingency plan, two communication relay plans, and a coverage plan. The contingency plan guides each robot to the element's goal location and was the same for all robots in the team. The two communication relay plans guide the robots to relay locations. Finally, a coverage plan guides each robot to the element's goal location but in a manner that sends the robots down different sides of the building—hence coming in two types. Thus, in total this element requires production of five internalized plans.

Currently the arrays produced by the decision tree set the gain for one plan to 1.0 and all others to zero. In future work we intend to blend advice from plans at each time step.

B. *Creating a Plan For Reconnaissance*

The process of determining which tasks decompose into the mission has already been discussed. Similarly, fleshing out a task into the decision tree and the parallel plans is currently an inexact and empirical process. The components of the `MoveTo` and `ReconBuilding` have been described in the preceding section. Next, the map locations that underlie each element's internalized plans are required. All positions are determined prior to running the mission and were, in this case, determined empirically.

The `MoveTo` element utilizes a parallel plan consisting of two internalized plans. The goal location for this element's contingency plan directs it to an arbitrary position outside the village. This is meant to regroup the team if communication fails when moving from one building to another. This element's other plan simply directs it to the element's goal position.

The `ReconBuilding` element employs a parallel plan consisting of four internalized plans. Because the contingency plan and both types of the coverage plan guide the robot to the same goal, only one goal location needs to be determined for these three plans. Additional positions are necessary for each of the two communication relay points, for a total of three positions.

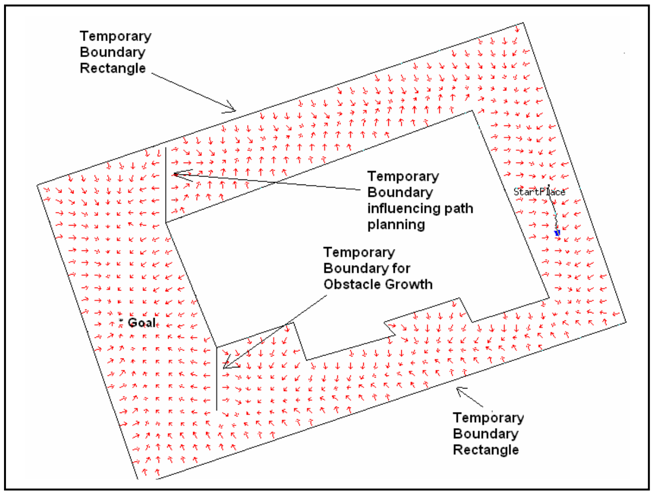

Temporary Boundary Rectangle
Temporary Boundary influencing path planning
StartPlace
Temporary Boundary for Obstacle Growth
Goal
Temporary Boundary Rectangle

Figure 7. An example of the vector field produced for the `ReconBuilding` plan element. Temporary boundaries guide the production of the vector gradient.

The different types of coverage have been mentioned briefly above. If one type leads the robot to the element's goal along a westerly (or northerly) path then the other type leads the robot along an easterly (or southerly) path. Constructing each type of coverage plan requires biasing the uniform cost algorithm to favor paths in one direction over paths in another direction. This is accomplished by adding cost to less favored path areas or by hallucinating a temporary boundary preventing access along one side of a building, which is the method we chose for ease of implementation. Figure 7 depicts a `ReconBuilding` element during construction. The gradient field directs the robot around the building from below. The temporary boundaries that are used to influence path direction and to grow the obstacle are marked. A temporary boundary that grows the building is employed to prevent communication attenuation at the far left corner of the building since the vector field slopes away from the communication relay robot. Both of these alterations to the internalized plan generation were used only for the two types of coverage plans in the `ReconBuilding` plan element.

In order to reduce plan computation time and the resulting resource file size, the internalized plans for the `ReconBuilding` element were not computed over the entire map. Rather a restricted rectangle was constructed around each building using temporary obstacles visible only to the internalized plan generation algorithm. These rectangles required four additional positions. Figure 7 shows the temporary boundary of the rectangle surrounding the building. In all, the generation and alteration of a single `ReconBuilding` element requires knowing fifteen positions: four points for the rectangle constructed around the building, four points marking the obstacle boundaries used to influence the coverage plans' path (two for each type/side), four points used to grow the building (two for each type/side), and three positions for the goals of the underlying plans.

After all the locations necessary for the reconnaissance mission have been collected, the individual internalized plans are compiled [13]. Upon completion, a resource file is generated. Software tools have been created to enable a user to alter specific internalized plans within a resource file. Individual plans can be added, deleted, or replaced without complete recompilation of all of the plans. Finally a mission is constructed employing the followplan behavior in the *MissionLab* behavior specification system.

C. *Utilizing a Reconnaisance Plan*

At startup, but prior to mission execution, the resource file containing data for each internalized plan is parsed and loaded into plan objects. At runtime, feature vectors are generated that include data produced by either a network model when running in simulation or a hardware network component developed by BBN Technologies and sent to the plan controller. The plan controller adds information pertaining to the size of the team and the robot's particular role in the plan. This information is provided to the current plan element. The plan controller selects which element to employ. This research explored using both a strict ordering of plan elements and a least commitment selection mechanism. Finally the element produces plan-based advice to be utilized by the robot controller or ignored. A task ends when all of the robots in the team have reached the element's goal location.

V. Experiments

Several experiments were run in simulation to test the new system. All experiments were run using the map and obstacle representation of a training village (figure 8). Experiments were conducted on the entire map and required thirteen `MoveTo` and thirteen `ReconBuilding` elements. The actual map contains 15 buildings. In two cases separate building were treated as a single complex due to their close proximity. These experiments consisted of thirty trials starting from a random location outside the urban terrain of the map. The robot teams were expected to navigate to and through the city. The map of the Fort Benning village is accurate to approximately one-meter resolution and covers a terrain of 220 square meters. Previously conducted tests have verified the ability of real robots to operate in and around the village with internalized plans using this map. The network model used for these experiments reduced network signal strength when the robots were occluded by terrain. Evaluation was based on total mission time, total distance traveled, urban terrain coverage, and percent time with at least one lone network. These metrics are felt to characterize performance in real world reconnaissance operations.

Estimates of baseline performance were obtained by comparing the plan element system to control experiments that used a three-robot team wandering the site and another using navigation based on *Missionlab*'s existing waypoint planner [14]. Although the control experiments do not provide a perfect comparison, they are meant to convey a general sense of the performance of uncoordinated reactive team behavior. All experiments used the same randomized start locations and identical gains for obstacle avoidance. The effect of team size was also investigated. As mentioned previously, this system operates for teams of up to four robots. Performance and

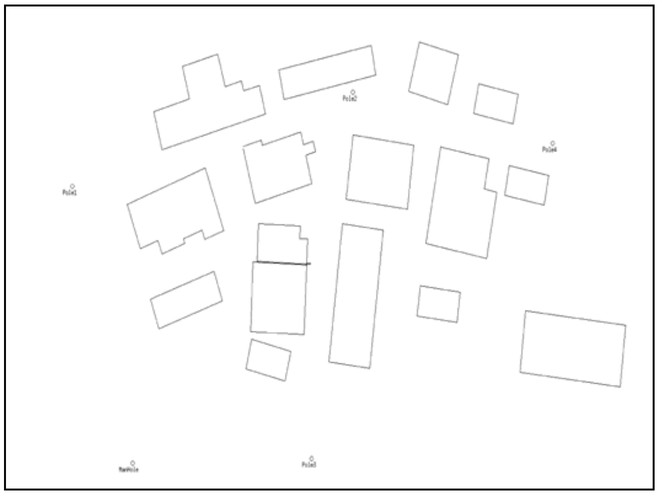


Figure 8. A map of the training village located at Fort Benning GA. All simulation trials were run on this test site. Early experiments with real robots have examined the validity of utilizing internalized plans at this site.

scalability were examined for two, three, and four robot team sizes.

We also experimented with different strategies for selecting plan elements. Even though plan elements are predefined in terms of their underlying locations, the plan controller selects elements dynamically at runtime for use by the robot. Using teams of three robots, a least commitment element selection mechanism was compared to selecting elements using a strict ordering determined by the experimenter. The least commitment mechanism consistently selected the plan element from the set of elements which minimized the distance from the current location to the element's goal location, thus continually selecting the nearest building for which reconnaissance had not yet been performed.

Finally experiments were performed using a scenario in which one robot of a three-robot team was constantly fixed (tethered) at a single location. This experiment explores an important real-world scenario where a human-operated vehicle deploys the reconnaissance robots that must explore an urban space while maintaining contact with the stationary base vehicle.

VI. Results

It was conjectured that by utilizing coordinated planning elements the percentage of lone or isolated networks would decrease significantly in comparison to the control systems. It was further hypothesized that as team size increases, network connectivity would improve. It was believed that as the total number of robots increases, although the task of coordinating becomes increasingly difficult, more opportunity for network connectivity exists. Figures 9-12 display the results for all experiments. Figure 9 examines communication attenuation. Due to lack of team coordination, the waypoint planner performed significantly worse then all other all other experiments on this metric ($p = 0$ on two-tailed t-test). Teams of three performed significantly worse then teams of other sizes ($p = 0$ vs. team size of two and team size of three). Teams of four maintained nearly perfect network connectivity with little variance over randomized start locations. The wander behavior

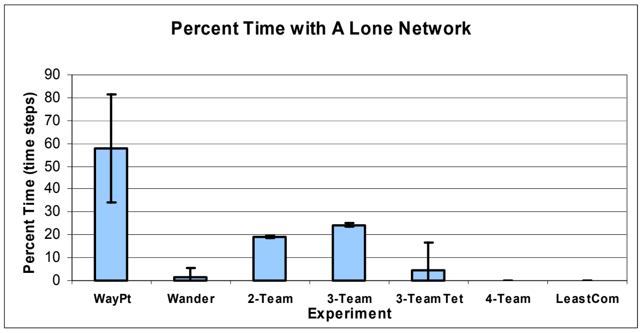


Figure 9. Percent time with a lone network. The waypoint control performs the worst. Teams using plan elements improves communication preformance.

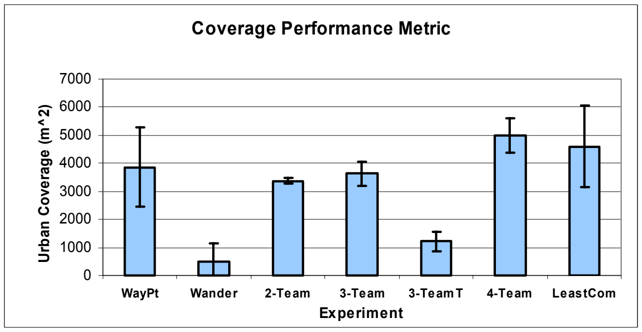


Figure 10. Coverage. The wander control performs the worst. Teams using plan elements cover large areas of urban terrain.

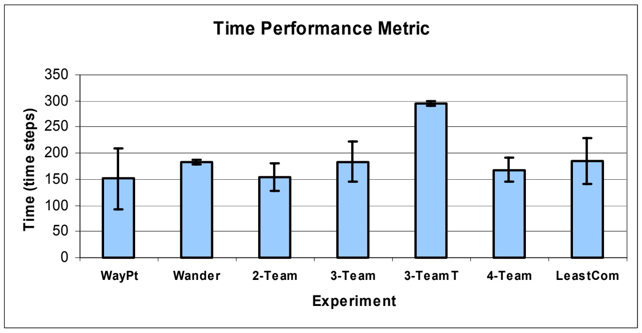


Figure 11. Time. Only the tethered team requires significantly greater time.

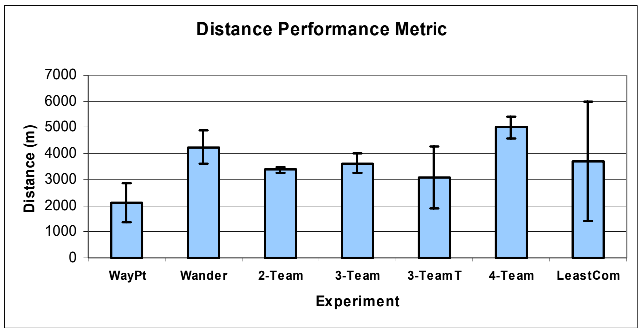


Figure 12. Distance—A measure of energy expenditure. When the plan controller utililizes a least commitment strategy the total distance traveled by a team of four robots deceases.

also performed well on this metric. This control's excellent communication results, however, reflect the behavior's lack of urban exploration rather then proper performance. More interestingly, the tethered team maintained a great deal of network connectivity resulting in a lone network only 4.6% of the time (*p =0*). This seems to indicate that a tethered team of robots could perform a probe-like reconnaissance mission in which communication maintenance was vital. The tethered robot team likely outperforms the untethered team because it has less opportunity for failure—it is tethered. Overall the communication results indicate the value of plan elements—coordinated planning reduces the number of isolated robots. We also see that additional teammates help maintain network connectivity, although this relationship is not strictly linear. Teams of three likely fare the worst because they tend to be unable to completely surround buildings and have two network connections to sever rather then one.

Figure 10 examines urban terrain coverage for each experimental condition. Greater coverage denotes better performance. Due to its lack of location coordinate the wander control performs the worst. As would be expected, the tethered team also performs poorly. Although the coverage performance of the remaining experiments occasionally differs significantly (*p =0* 3-Team vs. 4-Team), this is likely the result of our overly strict standard for coverage. In all experiments, it was assumed that each robot has only the ability to sense and hence cover a three by three meter area in all directions from its current location excluding obstacles. We suspect that relaxing this standard would result in approximately equivalent coverage for all experiments except the tethered experiment and the wander differences resulted. The tethered team's additional time is an experiment. Overall the coverage results indicate that large portion of the urban territory is being explored by the robots. Figure 11 displays the time required for each experiment. With the exception of the tethered team no significant indirect result of being tethered. These teams typically traverse most of the map, break communication, and then retreat back to the tethered robot. The back and forth nature of this scenario increases the time required to perform this mission.

Figure 12 depicts the total distance traveled by all robots in the team. This graph gives an idea of the energy requirements for the different types of experiments. The use of a least commitment strategy appears to have reduced the average energy consumption of a team of four robots to the equivalent of a team of three. As one would expect the variance of this strategy is large, however. Sometimes a least commitment approach to element selection works very well. Sometimes it works very poorly. However, on average this strategy performs well.

VII. Conclusions

This paper has presented a method for multi-robot communication-sensitive reconnaissance. More generally, it has outlined an approach by which larger tasks can be decomposed into smaller tasks and planned for by using internalized plans. Experiments using this method demonstrate improved performance over other control system experiments, and illustrate the effect of team size, plan selection, and scenario. Three interesting characteristics of this approach are worth noting:

1) Planning is offline in the sense that each plan element is computed a priori. This limits the planner to the elements that

have been created in advance but affords reactive utilization of each element.

2) Planning is online in the sense that the plan controller may select elements dynamically at runtime. This allows for dynamic reconfiguration of the plan and greater adaptability.

3) Several plans are utilized in parallel allowing the type of advice to be altered as environmental conditions change.

It could be claimed that simple iteration through the same series of waypoints by a team of robots would likely maximize our performance metrics. This is indeed the case. However, the primary aim of this work is the development of a technique suitable for battlefield scenarios. These situations demand the ability to deal with uncertain, unpredictable conditions and also utilize a priori information as much as possible. Equally important is the ability to react opportunistically to uncertainty. This approach is opportunistic in that plan elements can be managed dynamically; precompiled vector fields limit the tendency for unnecessary waypoints; and changes in the environment are reflected in output advice. Simple iteration through a series of waypoints is ill equipped to handle dynamic, unpredictable environments and lacks the ability to act opportunistically.

Currently, plan elements and their associated machinery are designed by hand. This limits the scalability of this approach with respect to the number of tasks. Scalability in terms of team size, terrain size, and instantiations of a particular task are not similarly limited. Hence this approach may be of value when the deployment situation could include arbitrary team sizes, terrain size, or repetitions of similar tasks.

In the future we hope to extend this approach to larger teams of robots, investigate more generalized and varied types of tasks, and examine the system's performance on real robots.

Acknowledgment

This research is funded under DARPA/DOI contract #NBCH1020012 as part of the MARS Vision 2020 program. This program is a joint effort of Georgia Institute of Technology, The University of Pennsylvania, University of Southern California, and BBN Technologies. We would like to thank the staff of the McKenna MOUT site at Fort Benning GA for allowing us to test our research at their facilities.

References

[1] R.C. Arkin and T. Balch, "AuRA: Principles and Practice in Review", *Journal of Experimental and Theorretical Artificial Intelligence*, 9(2):175-189, 1997.

[2] R.C. Arkin, *Behavior-Based Robotic*s, MIT Press, Cambridge, MA. 1998.

[3] T. Balch and R.C. Arkin, "Communication in Reactive Multiagent Robotic Systems," *Autonomous Robots*, 1(1):27-52, 1995.

[4] R.A. Brooks, "Intelligence without Reason," Artificial Intelligence, 47:139-159, 1991.

[5] J. Connell, "SSS: A Hybrid Architecture applied to Robot Navigation", *Proc. IEEE Intern. Conf. on Robotics and Automation*, pp 2719-2724, 1992.

[6] C.P. Diehl, M. Saptharishi, J.B. Hampshire II, and P. Khosla, "Colaborative Surveillance Using Both Fixed and Mobile Unattended Ground Sensor Platforms", *SPIE 13th International Conf. on Aerospace/Defense Sensing, Simulation, and Controls*, Vol. 3713, April, pp.178-185. 1999.

[7] E. Gat, "Three-Layer Architectures", in *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pp 195-210, MIT Press, Menlo Park CA, 1998.

[8] K. Konolige, "A Gradient Method for Realtime Robot Control", *Proc. IEEE Inter. Conf. on Inteligent Robotics and Systems*, pp 639-646, 2000.

[9] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[10] D.C. MacKenzie, "Design Methodology for the Configuration of Behavior-Based Robots", Ph.D. Diss., College of Computing, Georgia Inst. Of Tech., 1997.

[11] D.Payton, J. Rosenblatt, D. Keirsey, "Plan Guided Reaction*", IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1370-1382, 1990.

[12] A. Ranganathan and S. Koenig, "A Reactive Robot Architecture with Planning on Demand", *Proc. IEEE Inter. Conf. on Inteligent Robotics and Systems,* 1462-1468, 2003.

[13] A.R. Wagner and R.C. Arkin, "Internalized Plans for Communication-Sensitive Robot Team Behaviors", *Proc. IEEE Inter. Conf. on Inteligent Robotics and Systems*, pp 2480-2487, 2003.

[14] Arkin, R.C., "Navigational Path Planning for a Vision-based Mobile Robot, *Robotica*, Vol.7, pp.49-63, 1989.

Element Pseudocode

```
    Given feature vector V
    Initialize weight vector W =0
    W = DecisionTree(V)
    Advice vector $A = A_x + A_y$
    where
    $A_x = \sum_i^N w_i Q_{i_x}$
    $A_y = \sum_i^N w_i Q_{i_y}$ and
    $Q_i$ = vector from internal plan(i) from parallel plan
    return $A$
```

Plan Controller Pseudocode

```
    while Plan not complete
        retrieve feature vector from robot controller
        if element complete
            select new element using method m
        return advice vector A = Element(Feature vector)
```

Element selection Pseudocode

```
    if method m = least commitment
        Set current cost $= \infty$
        for all remaining elements in plan
            if element cost < current cost
                current cost = element cost
        return Element(current cost)
    else if method = strict ordering
        return Element(i)
```

Figure 13. Pseudocode for plan controller operation, element selection, and element advice production.