PROGRAMMING AND CONTROLLING THE OPERATIONS OF A TEAM OF MINIATURE ROBOTS *

Paul E. Rybski, Sascha A. Stoeter,
Maria Gini, Nikolaos Papanikolopoulos
Center for Distributed Robotics, Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455
{rybski,stoeter,gini,npapas}@cs.umn.edu

Abstract We describe a software architecture used to control the operations of a group of miniature mobile robots called Scouts. Due to their small size, the Scouts rely on a proxy processing scheme where they receive commands and transmit sensor information over RF channels to a controlling workstation. Because the bandwidth of these channels is limited, a scheduling system has been developed that allows the robots to share the bandwidth. Experimental results are described.

Keywords: Miniature robots, distributed sensing, resource sharing

1. Introduction

Tasks with multiple robots require a software framework in which behaviors can be easily integrated, and in which access to resources can be scheduled and managed by the controlling software without much user intervention. We have developed a distributed software system for controlling a group of small, mobile robots which have extremely limited on-board computing capabilities. These robots, called Scouts, are completely reliant upon a proxy processing scheme for all their computing needs, including the digitizing and processing of the video data they broadcast over a fixed-frequency analog radio link.

The communication channels the Scouts use to send and receive information are very limited in power and throughput. As a result, access

*Material based upon work supported by the Defense Advanced Research Projects Agency, Microsystems Technology Office (Distributed Robotics), ARPA Order No. G155, Program Code No. 8H20, issued by DARPA/CMD under Contract #MDA972-98-C-0008.

to these channels must be explicitly scheduled so that the demand for them can be met while maintaining the integrity of the system's operation. The Scout control architecture has been developed to take these factors into account.

We present experimental results on a distributed surveillance task in which multiple Scouts automatically position themselves in an area and watch for motion. We discuss how the limited communication bandwidth affects robot performance.

2. Scout Robots

Scouts are miniature (11.5 cm in length and 4 cm in diameter) robotic systems designed for surveillance and reconnaissance tasks (Rybski et al., 2000). They have a video camera which they use to transmit images to a remote source. They communicate over a packetized RF communications link using an ad-hoc networking protocol. Due to the Scout's limited volume and power constraints, the two on-board computers are only powerful enough to handle communications and actuator controls. All decisions and sensor interpretations are done on an off-board workstation or by a human teleoperator. Figure 1 shows a group of the robots.

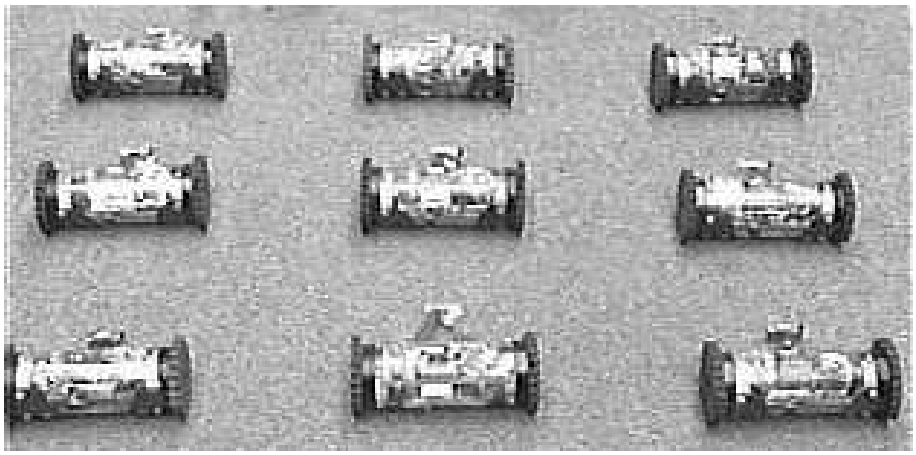

Figure 1. The fleet of Scout robots.

Video data is broadcast over a fixed-frequency analog radio link and must be captured by a video receiver and fed into a framegrabber for digitizing. Because the video is a continuous analog stream, only one robot can broadcast on a given frequency at a time. Signals from mul-

tiple robots transmitting on the same frequency disrupt each other and become useless.

The RF limitations of the Scout pose a couple of fundamental difficulties when trying to control several of them. First, the command radio has a fixed bandwidth. This limits the number of commands it can transmit per second, and therefore limits the number of Scouts that can be controlled simultaneously. Currently, we operate on a single carrier frequency, with a command throughput of 20-30 packets/second, which is sufficient to control 4 to 5 Scouts.

The most important problem is that there are not enough frequencies available in commercial off the shelf video transmitters to allow for a large number of simultaneous analog transmissions. With the current Scout hardware we have only two video frequencies. As a result, video from more than two robots requires interleaving the time each robot's transmitter is on. Thus, an automated scheduling system is required.

3. Dynamic Resource Allocation

We have designed a distributed software architecture (Stoeter et al., 2000), which dynamically coordinates hardware resources across a network of computers and shares them between client processes.

Access to physical hardware is controlled through components (software processes) called Resource Controllers (or RCs). If a decision process needs to use a resource, it must be granted access to its RC. Resources that can only be managed by having simultaneous access to groups of RCs are handled by a second layer components called Aggregate Resource Controllers (or ARC).

In order for a process to control a Scout, several physical resources are required. First, a robot not currently in use by another process must be selected. Next, a command radio with the capacity to handle the demands of the process is needed. If the Scout is to transmit video, exclusive access to a fixed video frequency is required, together with a framegrabber connected to a tuned video receiver.

Each instance of these four resources is managed by its own RC. In Figure 2 solid lines indicate which RCs the ARCs currently have access to. Dashed lines indicate RCs which are exclusive access only and can only support control from a single ARC. The Radio RC is an exception to this, as it is a sharable RC. Since ARC-2 has access to all of its RCs, it can run. ARC-1 cannot run because it is waiting on two RCs.

Access to RCs must be scheduled when there are not enough RCs to satisfy the requirements of the ARCs. A centralized process called the Resource Controller Manager maintains a master schedule of all

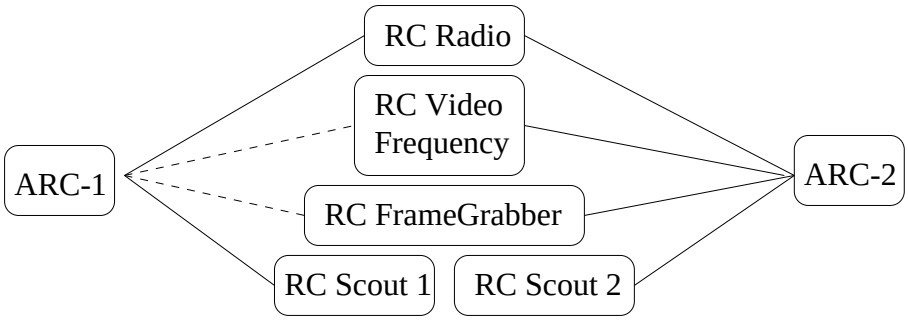

Figure 2. An example of how a decision process controls RCs by connecting to a single ARC.

active ARCs and grants access to each of their RCs when it is their turn to run. When requesting access to a set of RCs, an ARC must specify a minimum amount of time that it must run to get any useful work done.

The Resource Controller Manager's scheduling algorithm tries to grant simultaneous access to as many ARCs as possible. The ARCs that have some RCs in common are examined to determine which ARCs can operate in parallel and which are mutually exclusive. ARCs which request a non-sharable RC cannot run at the same time and must break their total operating time into slices. ARCs which have a sharable RC in common may be able to run simultaneously, assuming that the capacity requests for that sharable RC do not exceed its total capacity. Once the ARC schedule has been constructed, the Resource Controller Manager executes it and takes care of notifying the RCs which ARC they should talk to at any given point in the schedule.

4. Experimental Results

In the experiments the Scouts are used in a distributed surveillance task where they are deployed into an area and watch for motion. This is useful in situations where it is impractical to place fixed cameras because of difficulties relating to power, portability, or even the safety of the operator.

Several simple behaviors have been implemented to do the task. All the behaviors use the video camera, which currently is the only environmental sensor available to the Scout. These behaviors include Locate Goal which rotates the Scout in place while searching the area around it for a target area of interest, Drive Toward Goal which visually servos the robot to an area of interest, Handle Collisions which helps disengage

the Scout from an obstacle, and Detect Motion in which the Scout robot reports whether something in its field of view is moving.

To test the ability of the Scouts to operate in a real-world environment, a test course was set up in our lab using chairs, lab benches, cabinets, boxes, and miscellaneous other materials. The goal of each Scout was to find a suitable dark hiding place, move there, turn around to face a lighted area of the room, and watch for motion.

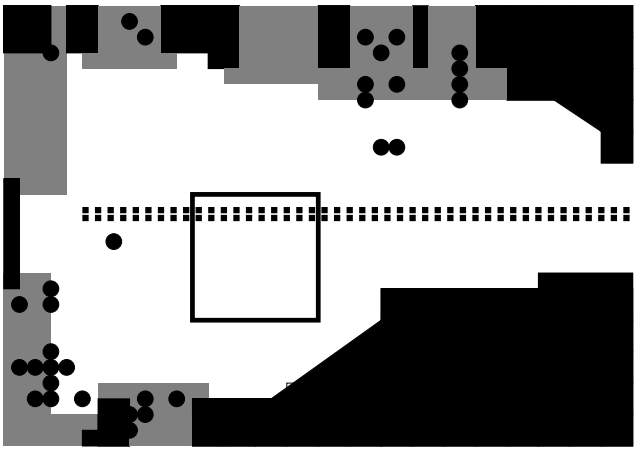

Figure 3. A top-down view of the room where the experiments were conducted. White areas are open space, gray areas are hiding spaces, and black areas are obstacles. The square outline in the center shows where the Scouts were started, the dotted line indicates the path of the moving target, and the dots are the hiding positions of the Scouts.

The environment, shown in Fig. 3, is 6.1 by 4.2 m and has a number of secluded areas in which the Scouts could hide. The Scouts were started at the center of the room and were pointed at one of 8 possible orientations. Both the position and orientation were chosen from a uniform random distribution.

The moving target the Scouts had to detect was a commercial mobile robot, chosen for its ability to repeatedly travel over a path at a constant speed. The target moved at a speed of approximately 570 mm/s and traversed the the room in 8.5 seconds on average. Once it had moved 16 feet into the room, it turned around and moved back out again. With a 4 second average turn time, the average time the target was in the room was 21 seconds.

When Scouts shared a single video frequency, only one Scout at a time could access the video frequency. Access was allocated and scheduled by the Resource Controller Manager. The amount of time each behavior could use the video frequency was set to 10 seconds, 3 seconds

of which were needed every time for the video-transmitter to warm up, so leaving 7 seconds for useful work.

Four different cases were tested: (1) a single Scout using a single video frequency, (2) two Scouts sharing a single video frequency, (3) two Scouts using two different video frequencies, (4) four Scouts sharing two different video frequencies. We run a total of 200 trials, with different hiding positions and number of scouts.

To evaluate the motion detection abilities of the Scouts and to determine the effect of sharing the video frequency, the actual time the target was seen (shown in Fig. 5) was compared to the potential time that the target could have been seen given the Scout positions (shown in Fig. 4). This potential time was calculated by analytically computing how long the target would be within the field of view of the Scout, independently on the state of activity of the Scout.

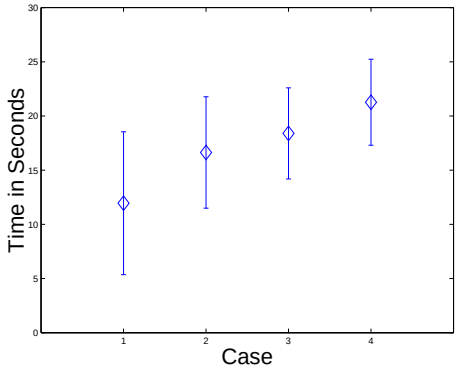

Figure 4. The potential time (in seconds) the Scouts could have been seen the moving robot. (1) one Scout, (2) two Scouts on a single frequency, (3) two Scouts on two different frequencies and (4) four Scouts on two different frequencies. Plots show means and standard deviations.

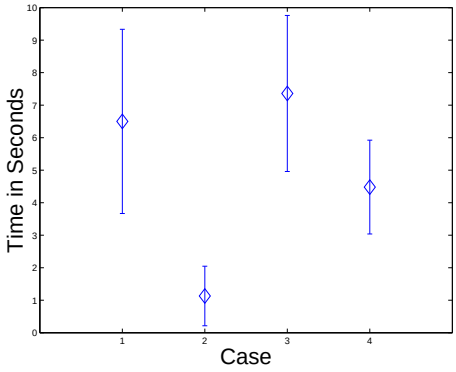

Figure 5. The actual time (in seconds) the Scouts detected motion. (1) one Scout, (2) two Scouts on a single frequency, (3) two Scouts on two different frequencies and (4) four Scouts on two different frequencies. The plots show means and standard deviations.

One Scout on a single frequency had a much higher success rate than two Scouts on a single frequency. This was expected because when the Scouts had to take turns with the video frequency, they could easily miss the target. The shorter the time the target was in the field of view, the smaller was the opportunity for the Scout to detect it, even when there was no swapping because a single frequency was used. Using a larger

number of Scouts increased the viewable area traversed by the target and the time that the target was in view, and decreased the variances.

The area viewed by four Scouts was significantly greater than the areas viewed in the other cases, but not by a factor of four over that viewed by one Scout nor by a factor of two over that viewed by two Scouts. The environment was such that there was usually a great deal of overlap in the areas viewed by individual Scouts. Redundancy was probably not as useful in this environment, but would probably be more effective in larger or more segmented environments. More details on the experiments are in (Rybski et al., 2001).

5. Related Work

Due to the small size, most miniature robots use proxy processing, as in Inaba *et al.* (Inaba et al., 1996), and communicate via a wireless link with the unit where the computation is done. This becomes a problem when the bandwidth is limited, as in the case of our Scouts. Because of their limited size, not only all processing is done off-board but also the communication is limited to a few communications channels.

A number of software architectures have been proposed for multiple robots, many of them described in (Kortenkamp et al., 1998). Our architecture has some similarities with ALLIANCE (Parker, 1998) and CAMPOUT (Pirjanian et al., 2000). The major difference is that we focus on resource allocation and dynamic scheduling, while other architectures are designed for more complex behavior fusion.

Resource allocation and dynamic scheduling are essential to ensure robust execution. Our work focuses on dynamic allocation of resources at execution time, as opposed to analyzing resource requests off-line, as in (Atkins et al., 2001; Durfee, 1999), and modifying the plans when requests cannot be satisfied. Our approach is specially suited to unpredictable environments, where resources have to be allocated in a dynamic way that cannot be predicted in advance. We rely on the wide body of algorithms that exists in the area of real-time scheduling (Stankovic et al., 1998) and load balancing (Cybenko, 1989).

6. Summary and Future Work

An essential feature of the distributed software control architecture we presented is the ability to dynamically schedule access to physical resources, such as communication channels and framegrabbers, that have to be shared by multiple robots.

We have also presented system issues related to the control of multiple robots over a low bandwidth communications channel. Experimental

results illustrating the ability of the Scout to position itself in a location ideal for detecting motion and the ability to detect motion have been shown. Future work is planned to allow the Scouts to make use of additional sensor interpretation algorithms for more complex environmental navigation. Ultimately, we hope to have the Scouts construct a rudimentary topological map of their surroundings, allowing other robots or humans to benefit from their explorations.

We believe that a combination of intelligent scheduling and more flexible hardware will allow a larger number of Scout robots to operate simultaneously in an effective manner.

References

Atkins, E. M., Abdelzaher, T. F., Shin, K. G., and Durfee, E. H. (2001). Planning and resource allocation for hard real-time, fault-tolerant plan execution. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):57–78.

Cybenko, G. (1989). Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel Distributed Computing*, 7(2):279–301.

Durfee, E. H. (1999). Distributed continual planning for unmanned ground vehicle teams. *AI Magazine*, 20(4):55–61.

Inaba, M., Kagami, S., Kanechiro, F., Takeda, K., Tetsushi, O., and Inoue, H. (1996). Vision-based adaptive and interactive behaviors in mechanical animals using the remote-brained approach. *Robotics and Autonomous Systems*, 17:35–52.

Kortenkamp, D., Bonasso, R. P., and Murphy, R. (1998). *Artificial Intelligence and Mobile Robots*. AAAI Press/MIT Press.

Parker, L. E. (1998). ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Trans. on Robotics and Automation*, 14(2):220–240.

Pirjanian, P., Huntsberger, T., Trebi-Ollennu, A., Aghazarian, H., Das, H., Joshi, S., and Schenker, P. (2000). CAMPOUT: a control architecture for multirobot planetary outposts. In *Proc. SPIE Conf. Sensor Fusion and Decentralized Control in Robotic Systems III*.

Rybski, P. E., Papanikolopoulos, N., Stoeter, S. A., Krantz, D. G., Yesin, K. B., Gini, M., Voyles, R., Hougen, D. F., Nelson, B., and Erickson, M. D. (2000). Enlisting rangers and scouts for reconnaissance and surveillance. *IEEE Robotics and Automation Magazine*, 7(4):14–24.

Rybski, P. E., Stoeter, S. A., Gini, M., Hougen, D. F., and Papanikolopoulos, N. (2001). Performance of a distributed robotic system using shared communications channels. Technical Report 01-031, Computer Science and Engineering Department, University of Minnesota.

Stankovic, J., Spuri, M., Ramamritham, K., and Buttazzo, G. (1998). *Deadline Scheduling For Real-Time Systems: EDF and Related Algorithms*. Kluwer Academic Publishers, Boston.

Stoeter, S. A., Rybski, P. E., Erickson, M. D., Gini, M., Hougen, D. F., Krantz, D. G., Papanikolopoulos, N., and Wyman, M. (2000). A robot team for exploration and surveillance: Design and architecture. In *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, pages 767–774, Venice, Italy.