Performance of a Distributed Robotic System Using Shared Communications Channels

Paul E. Rybski, Sascha A. Stoeter, Maria Gini, Dean F. Hougen, and Nikolaos Papanikolopoulos

***Abstract*—We have designed and built a set of miniature robots, called Scouts, and have developed a distributed software system to control them. This paper addresses the fundamental choices we made in the design of the control software, describes experimental results in a surveillance task, and analyzes the factors that affect robot performance.**

Space and power limitations on the Scouts severely restrict the computational power of their on-board computers, requiring a proxy-processing scheme in which the robots depend on remote computers for their computing needs. While this allows the robots to be autonomous, the fact that robots' behaviors are executed remotely introduces an additional complication – sensor data and motion commands have to be exchanged using wireless communications channels. Communications channels cannot always be shared, thus requiring the robots to obtain exclusive access to them.

We present experimental results on a surveillance task in which multiple robots patrol an area and watch for motion. We discuss how the limited communications bandwidth affects robot performance in accomplishing the task and analyze how performance depends on the number of robots that share the bandwidth.

***Index Terms*—Multiple robots, Mobile robots, Distributed software architecture, Resource allocation.**

I. Introduction

Controlling a group of miniature mobile robots in a coordinated fashion can be a very challenging task. The limited volume of miniature robots greatly limits the kinds of on-board computers and sensor processing systems they can use. One way to overcome these limitations is to use a communications link with a more powerful off-board processor. Unfortunately, the robots' small size also limits the bandwidth of their communications system and prevents the use of large capacity communications hardware (such as a wireless Ethernet). Scheduling access to the shared bandwidth becomes critical for effective operation.

We describe a case study of a group of miniature robots which must use very low capacity radio frequency (RF) communications systems due to their small size. The size limitations of these robots also restrict the amount of on-board computational power they can carry, forcing them to rely on off-board decision processes. Thus, all the sensor data are broadcast to a remote computer or a larger robot, and actuator commands are relayed back to the miniature robots. The operation of these robots is completely dependent on the RF communications links they employ. In order to handle high demand for this low capacity communications system, a novel process management/scheduling system has been developed.

Center for Distributed Robotics, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, U.S.A. (email: {rybski,stoeter,gini,hougen,npapas}@cs.umn.edu

In the experiments we describe, the robots are deployed to create a sensor network in an indoor environment and patrol the area watching for motion. We show how sharing bandwidth affects the performance of the robots when they are used in a surveillance task.

II. Miniature Robotic Systems

We have developed a set of small robotic systems, called Scouts [1], which are designed for reconnaissance and surveillance tasks. The Scout, shown in Figure 1, is a cylindrical robot 11.5 cm in length and 4 cm in diameter. Scouts locomote in two ways. They can use their wheels to travel over smooth surfaces (even climbing a 20° slope) and they are capable of jumping over objects 30 cm in height using their spring-loaded tails. Figure 2 shows the Scout jumping up a step.

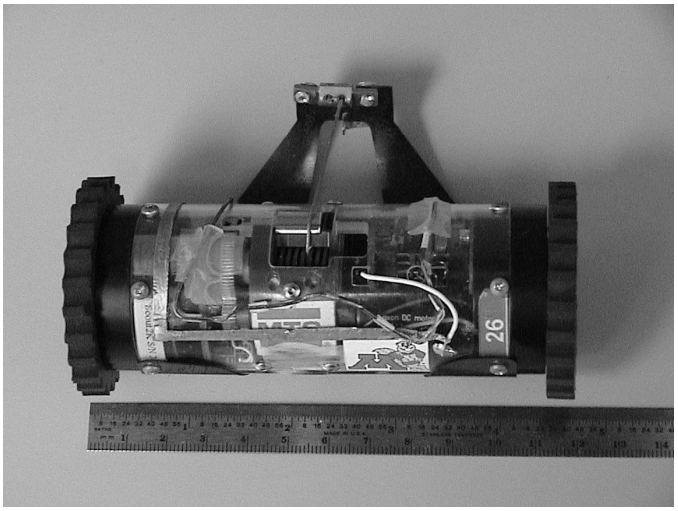


Fig. 1. The Scout robot shown next to a ruler (in cm) for scale.

The Scouts can transmit video from a small camera to a remote source for processing. They can also transmit and receive digital commands over a separate communications link that uses an ad hoc packetized communications protocol. Each Scout has a unique network ID, allowing a single radio frequency to carry commands for multiple robots. By interleaving packets destined for different robots, multiple Scouts can be controlled simultaneously.

Due to the Scout's limited volume and power constraints, the two on-board microprocessors are only powerful enough to handle communications and actuator controls. There is very little memory for any high-level decision process and no ability

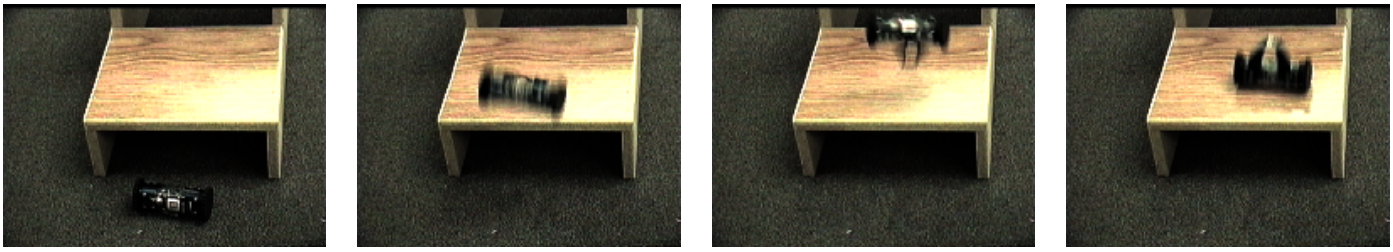


Fig. 2. A Scout using its spring-loaded tail to jump up a stair.

to process video. In order for the Scouts to accomplish anything useful, they must be paired with an off-board computer or a human teleoperator.

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 multiple robots transmitting on the same frequency disrupt each other and become useless. We will use the terms video frequency and video channel interchangeably throughout this paper.

The RF limitations of the Scout pose two fundamental difficulties when trying to control several Scouts. First, the command radio has a fixed bandwidth. This limits the number of commands it can transmit per second, and therefore the number of Scouts that can be controlled simultaneously. Currently, our inter-robot communications network operates on a single carrier frequency, with a command throughput of 20-30 packets per second.

Second, there are generally not enough commercial frequencies available to allow for a large number of simultaneous analog transmissions. With the current Scout hardware there are only two video frequencies available.[1] As a result, video from more than two robots can be captured only by interleaving the time each robot's transmitter is on. Thus, an automated scheduling system is required. Sharing the bandwidth among robots affects the performance, as we will see in the description of our experimental results in Section V.

III. Dynamic Resource Allocation

The decision processes that control the actions of the Scouts need to be able to connect to all the resources necessary to control the physical hardware. We have designed a software architecture [2] which connects groups of decision processes with resource controllers that have the responsibility of managing the physical resources in the system.

This distributed software architecture dynamically coordinates hardware resources transparently across a network of computers and shares them between client processes. The architecture includes various types of user interfaces for robot teleoperation and various sensor interpretation algorithms for autonomous control. The architecture is designed to be extremely modular, allowing for rapid addition of behaviors and resources to create new missions.

[1] This was true when this paper was originally written. Since then, the number of available commercial frequencies has increased to six.

Access to robotic hardware and computational resources is controlled through processes called *Resource Controllers* (RCs). Every physical resource has its own RC. Any time a behavior or another decision process needs a particular resource, it must be granted access to the appropriate RC. Some physical hardware can only be managed by having simultaneous access to groups of RCs. This grouping is handled by a second layer called *Aggregate Resource Controllers* (ARCs). Every ARC is an abstract representation of the group of RCs that it manages. An ARC provides a specialized interface into the group of RCs that it manages.

A. An Example of ARCs and RCs

In order for a process to control a single Scout, several physical resources are required. First, a robot which is not currently in use by another process must be selected. Next, a command radio which has the capacity to handle the demands of the process is needed. (Refer to Section III-C for a discussion of the radio's capacity.) If the Scout is to transmit video, exclusive access to a fixed video frequency is required, as well as a framegrabber connected to a tuned video receiver. Each instance of these four resources is managed by its own RC.

Figure 3 illustrates the interconnections between the components in the system. In this example, a hierarchy of behaviors is responsible for controlling two robots and a user interface teleoperation console lets a user control a third. Each component has its own ARC which attempts to gain access to the appropriate resources. There are three Scout robots, all of which share a single video frequency. A single video receiver is attached to a video processing card, and a Scout command radio is attached to a serial port. The ARCs must share the video frequency and framegrabber RCs. The ARC owned by the teleoperation console does not need the framegrabber but still needs control of the video frequency to operate. In this situation, only one of the three ARCs will be able to send commands to its robot at a time and thus the ARCs must have their access scheduled.

B. The Resource Scheduler

Access to RCs must be scheduled when there are not enough RCs to satisfy the requirements of the ARCs. The central component which oversees the distribution and access to the ARCs and RCs is the Resource Controller Manager. The Resource Controller Manager maintains a master schedule of all 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

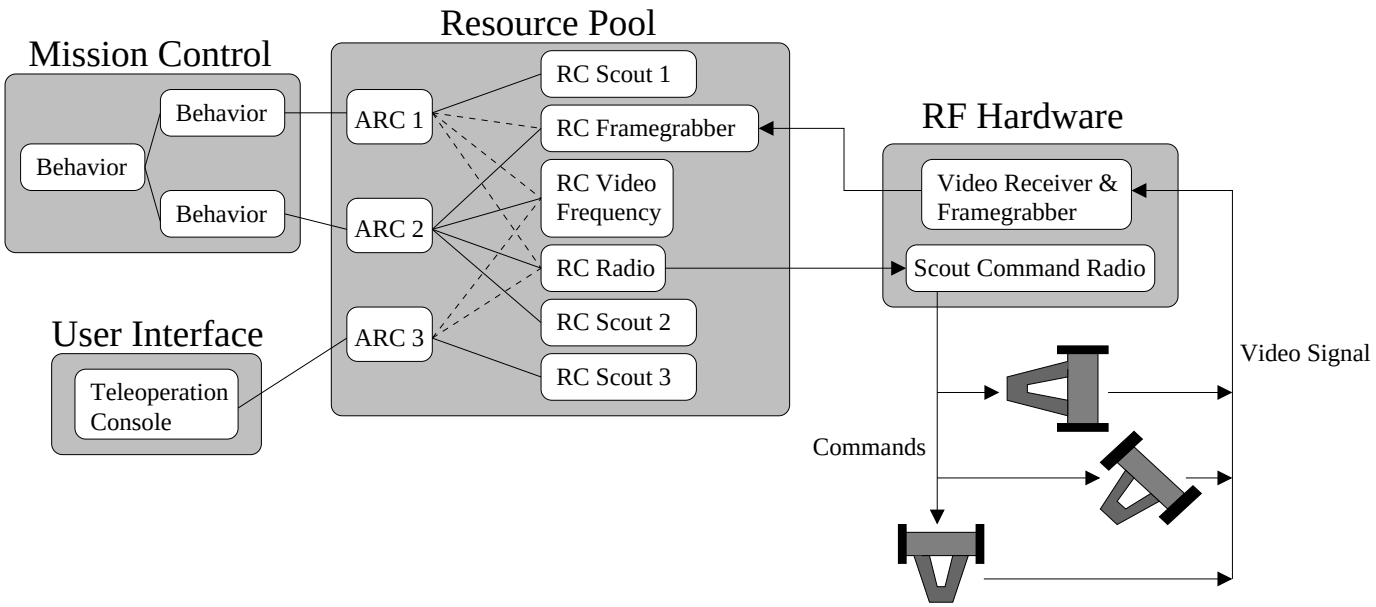


Fig. 3. An instance of the architecture. Three Scouts are controlled by a combination of behaviors and a teleoperation console. All three share the same video frequency, so only one robot can be controlled at a given time. Solid lines indicate active connections (where data can flow between components) while dashed lines indicate connections that are not currently active but may become active later.

that it must run to get any useful work done. This value, which is generally on the order of seconds to minutes, is called the minimum runtime value.

The RESOURCE CONTROLLER MANAGER's scheduling algorithm tries to grant simultaneous access to as many ARCs as possible. ARCs are divided into sets depending on the RCs they request. ARCs that ask for independent sets of RCs are put into different groups. These groups will run in parallel with each other since they do not interact in any way. 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.

ARCs with higher priorities are given precedence over ARCs with lower priorities. The RESOURCE CONTROLLER MANAGER attempts to generate a schedule which allows all ARCs of the highest priority to run as often as they are able to. Any ARC of a lower priority which can run at the same time without increasing the wait time of any of the higher-priority ARCs, is also allowed to run. Lower priority tasks that cannot be so scheduled must wait (possibly indefinitely) for the higher priority tasks to complete.

Once the schedule has been generated, the schedule manager iterates over it in a simple round-robin fashion. ARCs are started and stopped according to the length of their minimum runtimes until the controlling behaviors request that they be removed from the schedule, at which point the RESOURCE CONTROLLER MANAGER recalculates the new schedule. As an example, in Figure 3, the sequence that the three ARCs would be run in is *(1,2,3,1,2,3, . . .)*. For a more detailed discussion on the scheduling algorithm as well as examples, please refer to [3].

C. Sharable Resources

Sharable RCs, such as the Scout radio, have to manage their own schedules to ensure that each of the ARCs using them is given a chance to send packets to their robot at their requested rate. When requesting access to a sharable RC, an ARC must specify a usage parameter which defines how often it will make requests and, if relevant, what kinds of requests will be made. In order to streamline the scheduling process, commands sent to sharable RCs must have a constant interval between invocations. In addition, each request must be completed before the next request is made. However, because the CPU load of any given computer will vary depending on how many components are running on it, the run-time of any given request may vary.

Sharable RCs use a simple rate monotonic algorithm (RMA) [4] to schedule access. Other more complex algorithms could be used, such as the algorithm for proportional share resource allocation [5] or the algorithm proposed in [6] for fair allocation of a single resource to a set of tasks. In our system, we rely on user-set priorities for sharing resources, so we are not as concerned about fairness and more concerned about efficiency and simplicity.

Requests with higher frequencies have precedence over requests with lower frequencies. Once again, however, the user-set priorities must be maintained. Thus, higher user-set priority ARCs have precedence over lower user-set priority ARCs regardless of the frequency of the requests. This can cause a disruption in the way requests are handled by the scheduling algorithm and may produce a schedule which is suboptimal in its usage of the RCs. Only when all of the higher-priority RCs have been scheduled will the lower priority RCs be allowed access. If a sharable RC cannot schedule all the ARCs, the responsibility for handling requests is given to the RESOURCE CONTROLLER MANAGER.

Once the requests for access have been granted, the ARCs

can use them in any way they see fit. Until they make a request for a specific command to be sent to the radio, for instance, the timeslices devoted to those ARCs are empty and the radio does nothing.

As illustrated in Figure 4, several ARCs have been granted access to a radio, which is a sharable RC. ARC1 has requested one half of the available bandwidth and thus is given every other timeslot. ARC2 and ARC3 have requested one quarter and one eighth of the available bandwidth respectively. There is still enough bandwidth for another ARC to request the remaining one eighth of the available bandwidth of this RC. This schedule could not exist in the example shown in Figure 3 because those ARCs cannot run simultaneously. If they each had their own Video Frequency and Framegrabber ARCs, then this Radio RC schedule would be possible.

Time	ARC ID
1	ARC1
2	ARC2
3	ARC1
4	ARC3
5	ARC1
6	ARC2
7	ARC1
8	Empty Slot

Fig. 4. A typical schedule for a sharable RC such as a Radio RC showing what timeslots are available to the scheduled ARCs.

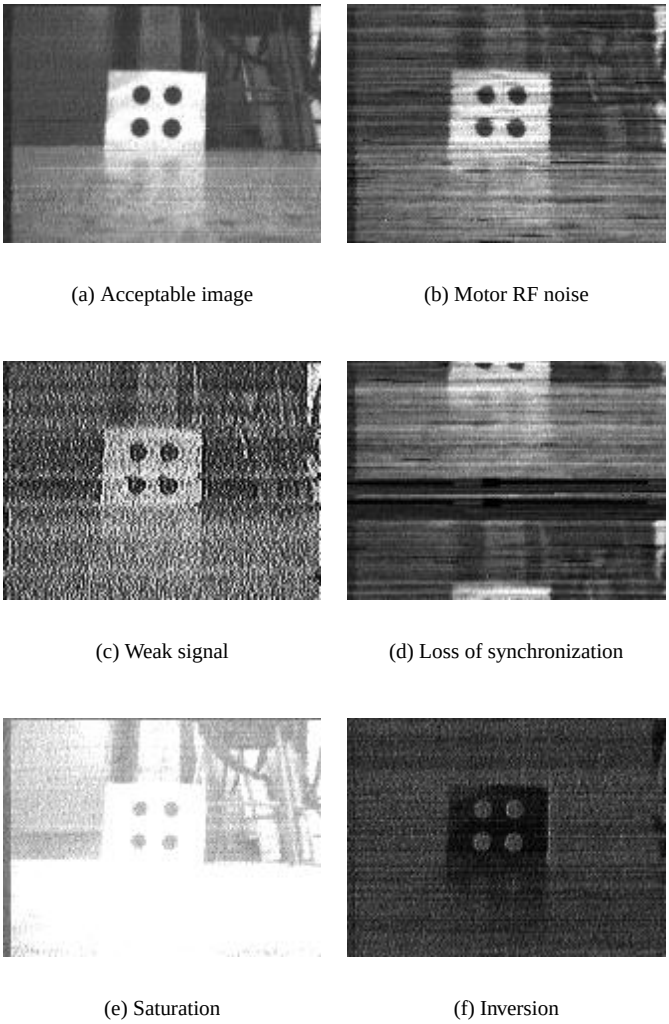
IV. A Distributed Surveillance Task

The Scouts are used in a distributed surveillance task where they are deployed into an area to 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. In this task, the Scouts can either be deployed into their environment by a human or another robot, or they can autonomously find their way into useful areas.

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. Using the video camera presents several problems. One problem is the Scout's proximity to the floor, which severely restricts the area it can view.

Since the video is broadcast over an RF link to a workstation for processing, its quality often degrades due to noise caused by the Scout's motors, multi-path reflections caused by obstacles around the robot, or weak signals caused by proximity to the ground and excess distance between transmitter and receiver. Figure 5 illustrates how noise can affect the quality and clarity of returned images.

In earlier work, we used a simple frame averaging algorithm to reduce the effects of noise [7]. This approach only dealt with the problem of spurious horizontal lines and white noise (Figures 5(b) and 5(c), respectively). If the image became saturated/inverted (Figures 5(e) and 5(f), respectively), or if vertical synchronization was lost (Figure 5(d)), averaging only compounded the problem.



(a) Acceptable image (b) Motor RF noise

(c) Weak signal (d) Loss of synchronization

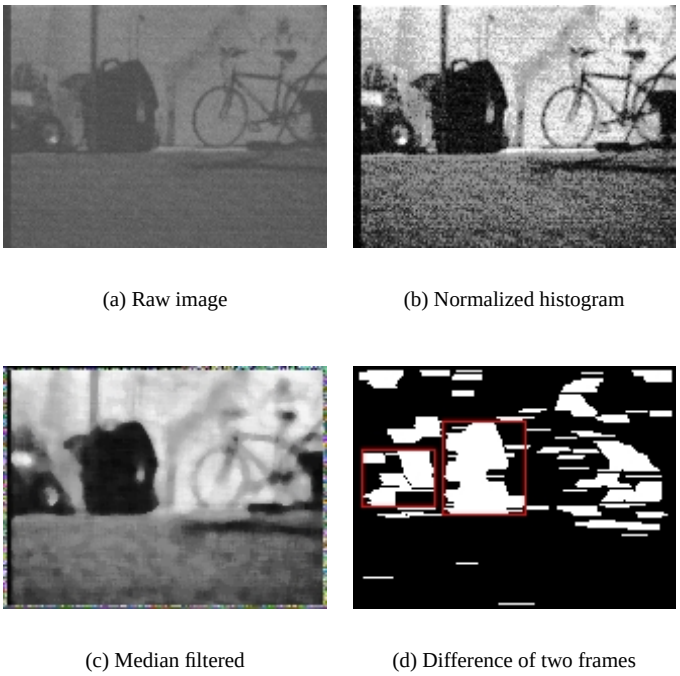
(e) Saturation (f) Inversion

Fig. 5. Effects of RF noise in Scout video.

Currently, the grayscale histogram of the Scout video is normalized (Figure 6(b)) in order to accentuate the contrasts between light and dark areas. However, this has the side effect of enhancing RF noise. To compensate, we apply a 5×5 median filter (Figure 6(c)) over the image to smooth the data. The median filter is faster than applying a Gaussian convolution mask and does a fairly good job of removing much of the most common noise. We have implemented several heuristic filters to remove the data corrupted by RF noise. These filters were generated by hand after analyzing how the video is corrupted by RF noise. Often, when the video transmitted from a Scout is severely corrupted, the best choice to reduce the noise is to reposition the Scout. Figure 6(d) shows the result of performing frame differencing and connected region extraction. This image was generated by rotating the Scout counterclockwise in place. The white regions are connected regions that are different between the two images. The two gray rectangles highlight blobs that are likely caused by the Scout's motion. All of the other blobs are considered caused by random RF noise (the decision depends on their shape and area) and are thus ignored.

The behaviors we have implemented for this task are:

Locate-Goal: Determining the location of the darkest (or light-



(a) Raw image (b) Normalized histogram

(c) Median filtered (d) Difference of two frames

Fig. 6. Scout image processing algorithms.

est) area of the room is accomplished by spinning the Scout in a circle and checking the mean value of the pixels in the image. The circular scan is accomplished in a number of discrete movements. The Scout captures an image, rotates for half a second, takes a new image, and subtracts the new image from the old one. A large difference in the images indicates the Scout moved. This approach can fail if the image quality is so low that motion in the image cannot be distinguished from noise. If the robot is operating in an area of very low light or uniform color, there may not be enough detail in the images to generate significant differences. Normalizing the histogram, as described earlier, helps to increase the contrast between different objects in the image, allowing the objects to stand out when the Scout moves.

Drive-Toward-Goal: Identifying a dark area to move toward is a simple matter of analyzing a strip in the image along the horizon and determining the horizontal position of the darkest area. The Scout computes the darkest region and tries to servo in that direction. The Scout will stop when its camera is either pressed up against a dark object, or if it is in shadows. If either of these two methods fail, this behavior will time out and quit after a minute or two of operation. Scout motion in this behavior is continuous and the Scout does not check its movements by frame differencing because it does not move very quickly. The difference between subsequent frames captured during forward motion is often minimal, making it difficult for the Scout to detect its own motion.

Detect-Motion: Detecting moving objects is accomplished using frame differencing. The Scout stays still and subtracts sequential images in the video stream and determines whether the scene changes at all (caused by movement in the image.) RF noise can also cause a great deal of perceived motion between frames. This is filtered out by analyzing the shapes and sizes of the blobs and ignoring blobs that are caused by noise. Currently, a hand-tuned filter is used for this decision process.

Handle-Collisions: If the Scout drives into an obstacle, all motion in the image frame will stop. If no motion is detected after the Scout attempts to move, it will invoke this behavior and start moving in random directions in an attempt to free itself. In addition to freeing the Scout, this random motion has the additional effect of changing the orientation of the antenna, which might improve reception.

V. Experimental Results

The Scouts' ability to accomplish the surveillance task was examined with a series of experimental runs. These experiments were designed to test the individual and team performances of the Scouts and the controlling architecture in a number of different situations and group configurations. In particular, we were interested in evaluating:

- the effectiveness of the vision-based behaviors for navigating the Scouts to useful positions;
- the performance of the scheduling system with multiple Scouts using the limited bandwidth RF video channels to detect motion;
- the performance of the robotic team given a set of specific constraints for the system and the environment.

Three experiments were run to evaluate the Scouts' performance:

A) Visual Servoing. A Scout has to locate a dark area and move to it.
B) Hiding and Viewing a Room. One or more Scouts have to hide in a dark area and turn to view a bright area.
C) Detecting Motion. One or more Scouts have to detect a moving object.

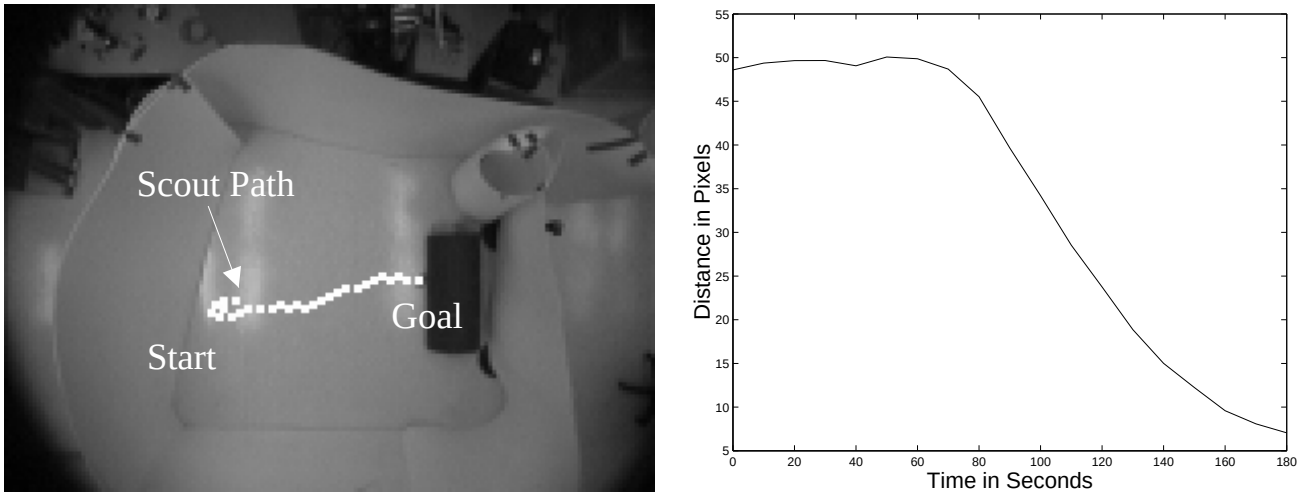


Fig. 7. Experiment A: Visual Servoing. Validation of the Locate-Goal and Drive-Toward-Goal behaviors. The left image shows a top-down view of the experiment. The right image shows the average distance in pixels of the Scout from the target.

A. Visual Servoing

An initial experiment was done to determine how well the Scout could locate and move to an area, using images from its camera. The environment consisted of a roughly $2.5\,\mathrm{m} \times 3\,\mathrm{m}$ enclosure with uniformly-colored walls and a $1\,\mathrm{m} \times 0.5\,\mathrm{m}$ black rectangle on one side of the enclosure as the target for the Scout. The Scout was started $1.5\,\mathrm{m}$ away from the center of the target. This experiment was designed to examine the Scout's Locate-Goal and Drive-Toward-Goal behaviors.

Nine trials were run to see how long it would take the Scout to locate the black target object and move to it. A camera was mounted on the ceiling of the room and was used to view the progress of the Scout from above. This camera was used for data logging purposes only. The Scout did not use this video data to navigate with. A simple tracking algorithm was used to automatically chart the progress of the Scout as it moved toward the target. Figure 7(a) shows the view from the overhead camera as well as a superimposed plot of the path that the Scout took to reach its objective during one of its nine trials. In each case, the Scout successfully located the target and moved to it.

Figure 7(b) shows a plot of average distance from the Scout to the target vs. time for all of these trials. In the first 70-80 seconds, the Scout used its Locate-Goal behavior to find the dark spot. Once a suitable spot was located, the Scout used its Drive-Toward-Goal behavior until it came in contact with the goal, somewhere between 150 and 160 seconds after the start of the trial.

B. Hiding and Viewing a Room

To test the ability of the Scouts to operate in a more 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 in these experiments was to find a suitable dark hiding place, move there, and turn around to face a lighted area of the room.

The environment, shown later in Figure 8, was $6.09\,\mathrm{m}$ by $4.26\,\mathrm{m}$ and had a number of secluded areas in which the Scout could hide. The test course had $13.48\,\mathrm{m}^2$ of open space, $7.99\,\mathrm{m}^2$ of obstructed space, and $4.55\,\mathrm{m}^2$ of potential hiding places. The Scouts were started at the center of one of the 16 tiles (each of which is $0.09\,\mathrm{m}^2$) in the center of the room and were pointed at one of eight possible orientations. Both the position index and orientation were chosen from a uniform random distribution.

The hiding and viewing experiment was divided into three cases, each using a different number of Scouts or communications channels. Within each case, ten trials were run. The stopping positions and orientations of the Scouts from the end of the trials were used later for the detect motion experiment (Experiment C). In the first case, a single Scout on a single video frequency was used to serve as a baseline. The second case used two Scouts that had to share a single video frequency. The third case used two Scouts, each on its own video frequency.

When Scouts shared a single video frequency, access to the video frequency was scheduled by the RESOURCE CONTROLLER MANAGER. For these experiments, each Scout's behavior requested ten second intervals of access to the video frequency. However, since the video transmitter requires 2-3 seconds of warm-up time before the image stabilizes, Scouts effectively had only seven seconds of useful viewing time when they were granted access.

Figure 8 shows the hiding places found for all trials. Over all the trials, the Scouts were able to hide themselves $90\,\%$ of the time. In the remaining $10\,\%$ of the time, the Scouts reached a 60 second time-out on the Drive-Toward-Goal behavior, and stopped out in the open where they could be more easily seen and bumped into. This time-out was required because the Scouts are unable to determine with confidence if progress is being made in moving toward a hiding position. This time-out was also encountered on some successful hiding trials, as the Scout continued to try to progress to a darker hiding position, even after reaching cover. For this reason, the non-hiding trials are not considered outliers in the time data.

Once the Scouts had positioned themselves, they attempted to orient themselves to view a lighted area of the room. Figure 9 shows the times for the Scouts to reach their final poses (positions and orientations). In the cases with two Scouts the

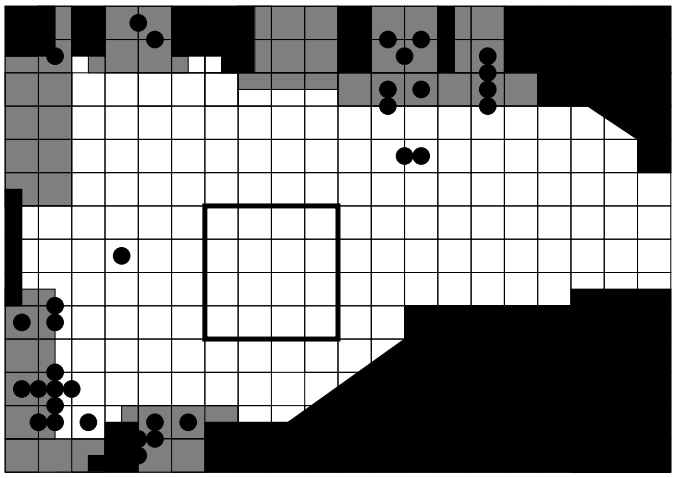


Fig. 8. Experiment B: Hiding and Viewing. Positions that the Scouts found for themselves in the 6.09 m by 4.26 m room are represented as dots.

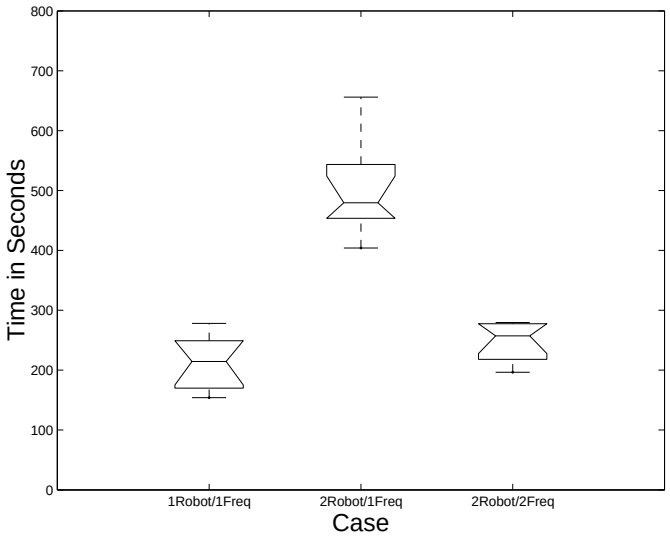


Fig. 9. Experiment B: Hiding and Viewing. The average time that it took the Scouts to complete each trial, shown for the three different cases. The averages are over 10 trials per case. The times are plotted using a box representation where the center line is the median value, the top and bottom lines of the box represent the upper and lower quartile values respectively, and the lines at the top and bottom of each plot represent the rest of the distribution. The notches in the box represent an estimate of the distribution's mean.

value plotted is the average time. As can be seen from the figure, two Scouts on a single video frequency took longer to reach their final poses than a single Scout. This is to be expected—the Scouts are time-multiplexing the video frequency resource. There is also a somewhat greater average time for two Scouts on two different video frequencies to reach their final poses than there is for the single scout case (for the first case , mean = 212.50, σ = 44.55; for the third case, mean = 247.50, σ = 30.62), however, these differences are not statistically significant at the 95 % confidence level (two-tailed, two-sample t test, p = 0.0555).

One interpretation of these results is that one Scout is better than two on the same frequency (as the task is accomplished more quickly by one) and that one Scout and two on different frequencies are approximately equal on this task. However, this ignores the fact that two Scouts can potentially accomplish more than a single Scout.

Nonetheless, even if two Scouts could accomplish twice as much as one after reaching their final poses, one Scout is still better, on average, than two on the same frequency. The time two Scouts spent hiding is significantly greater than twice the time one Scout spent. This is because when switching cameras up to 30 % of the time is lost waiting for the video transmitter to warm up. For this reason, deploying two Scouts sequentially would make more sense than deploying them in parallel if the Scouts must share the video frequency. An instant-on transmitter would eliminate this advantage for sequential deployment.

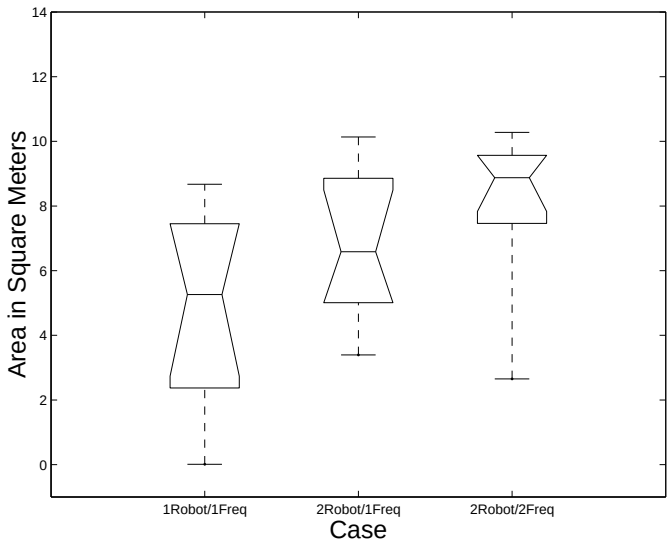


Fig. 10. Experiment B: Hiding and Viewing. The total areas that the Scouts were able to view.

Since the overall mission is surveillance, one measure of Scout performance after deployment is the open area viewed. Figure 10 shows the total area viewed by the Scouts for each case. Considering the area viewed, two Scouts on different frequencies are better than one, as the area viewed is larger (for one Scout, mean = 4.73, σ = 2.89; for two Scouts with two frequencies, mean = 8.09, σ = 2.34)—this difference is significant (one-tailed, two-sample t test, p = 0.0053).

C. Detecting Motion

A third experiment was run to test the Scouts' detect motion abilities. Four different cases were tested, including a single Scout using a single video frequency, two Scouts sharing a single video frequency, two Scouts using two different video frequencies, and four Scouts sharing two different video frequencies.

For each of the four cases, the Scouts were placed in ten different positions in the environment. These positions were the same as the hiding positions obtained in the previous experiment. In the case using four Scouts, for which no hiding experiment was run, the positions were randomly sampled with replacement from the results of the other hiding experiments. In each position, five individual motion detection trials were run, bringing the total number of individual trials to 200.

In these experiments, the video frequency was swapped between the robots every eight seconds and the delay for the camera to warm up was set to four seconds. These values were chosen to give the Scouts a lower latency between observations. The longer warm-up time was necessary because the detect motion behavior is more sensitive to noise than the navigation behaviors.

The moving target the Scouts had to detect was a Pioneer 1 mobile robot [8]. A Pioneer was chosen for its ability to repeatedly travel over a path at a constant speed. This reduced some of the variability between experiments that the use of human subjects might have caused.[2] The Pioneer entered the room from the right and made its way over to the left, moving at a speed of approximately $0.57\,\mathrm{m/s}$ and traversing the room in 8.5 seconds on average. Once it had moved $4.87\,\mathrm{m}$ into the room, it turned around and moved back out again. With a 4 second average turn time, the Pioneer was in the room on average for 21 seconds.

Figure 11 illustrates the fields of view seen by two Scouts and the area of the Pioneer's path that they cover. While the views of these Scouts do not overlap, there was a large amount of overlap in some of the other placements of Scouts.

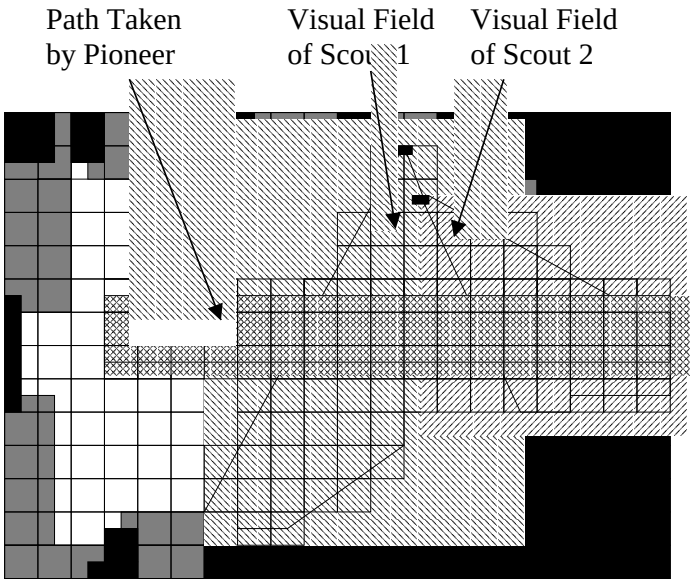


Fig. 11. Example Scout placement in the room. In this instance, there are two Scouts that view the path of the Pioneer robot, shown in dark gray in the middle of the room. The fields of view of the two Scouts do not happen to overlap.

Figure 12 shows the total areas viewed by the Scouts in each of the four cases. The area viewed by four Scouts is significantly greater (at the $95\,\%$ confidence level) 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 size and configuration of 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 (two or three Scouts might have sufficed), but would probably be more effective in larger or more segmented environments.

[2] Additional experimentation showed that the Scouts were at least as good at detecting human motion as they were at detecting the Pioneer.

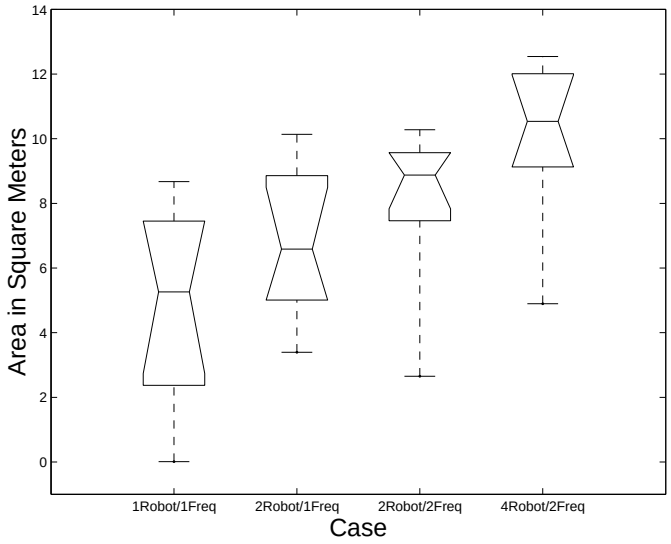


Fig. 12. Experiment C: Detecting Motion. The areas that the Scouts were able to view. Averages are computed over 50 trials, five trials for each of the ten positions of the Scouts.

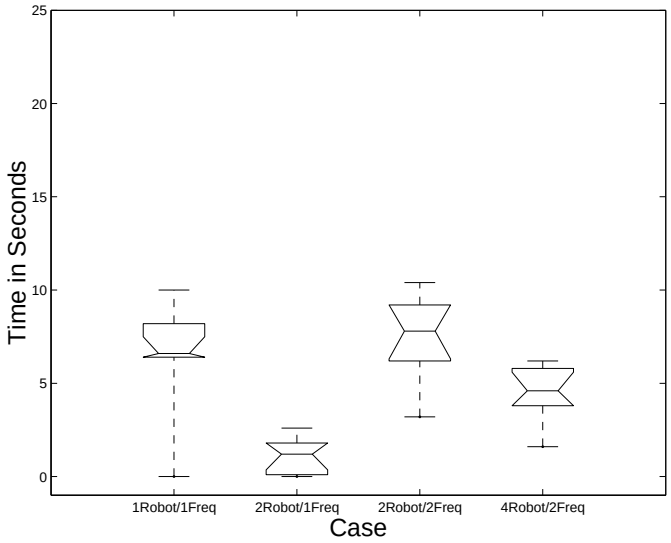


Fig. 13. Experiment C: Detecting Motion. The actual time that the Scouts detected the motion of the Pioneer.

Figure 13 shows the total amount of time the target was seen by the Scouts in each of the four cases. There are two major factors that affect the performance of the Scouts at detecting motion: (1) the nature of the detect motion algorithm, and (2) the sharing of the bandwidth.

The detect motion algorithm is sensitive to the distance of the moving object from the Scout and to the direction of movement with respect to the optical axis of the Scout. When the Pioneer moved perpendicularly across the Scout's optical axis, the Scout was able to detect it easily. However, when the Pioneer moved parallel to the optical axis, the Scout had a difficult time detecting it. This is due to the relatively small change between successive video frames. For the same reason movements of objects farther away are harder to detect. More details on this are in Section VI-A.

To make more explicit the effect of sharing the video fre-

quency, we show the actual time that the Pioneer was seen compared to the potential time it could have been seen. By plotting the measured target detection time for the cases using a single frequency (see Figure 14) and for the cases using two frequencies (see Figure 15, we see clearly how sharing bandwidth reduces performance.)

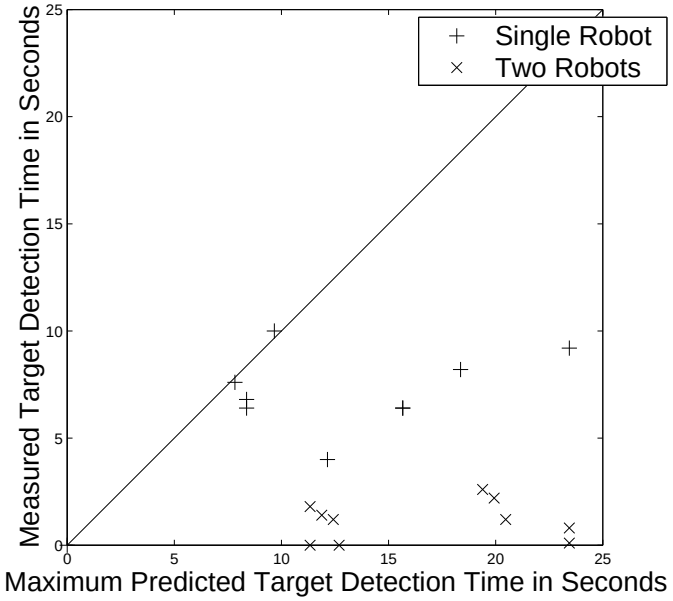


Fig. 14. Experiment C: Detecting Motion. Single frequency cases. The horizontal axis represents the maximum possible time the Pioneer could be detected by the Scouts and the vertical axis represents the time it actually was. The closer these two values are, the better the performance.

Figure 14 shows the plot of the cases with a single frequency using one and two Scouts. As can be seen, the one Scout case had a much higher success rate than the two Scout case. This was expected because the robots in the two Scout case were not able to view the entire area at one time. Since they had to share a video frequency, they had to take turns observing their respective fields of view. Since the Pioneer was moving relatively quickly (over half a meter a second), it would be missed if the Scout did not have access to the video frequency at that time.

Figure 15 shows the actual time the Pioneer was detected compared to the potential time it could have been detected for the experiments with two and four Scouts using two frequencies.

To complete our analysis, we need to account for an additional factor. The area traversed by the Pioneer that was visible to the Scouts and the amount of time the Pioneer was visible were different across experiments. This was caused by the fact that the Scouts did not always hide in the best viewing positions. In some experiments, one Scout was facing the wall instead of facing the open area, and so it did not contribute to the detection task at all. In other cases, two Scouts were very close with their viewing areas almost completely overlapping.

Figure 16 and Figure 17 show respectively the area traversed by the Pioneer that was in the field of view of the Scouts and the time the Pioneer was in the field of view of the Scouts for the different experiments. This gives an indication of the complexity of the task. The smaller the area and the shorter the time, the

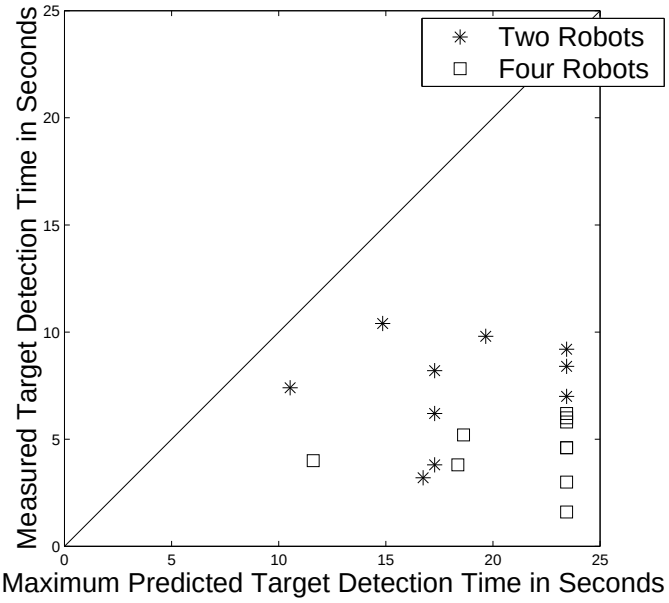


Fig. 15. Experiment C: Detecting Motion. Double frequency cases. The horizontal axis represents the maximum possible time the Pioneer could be detected by the Scouts and the vertical axis represents the time it actually was. The closer these two values are, the better the overall performance.

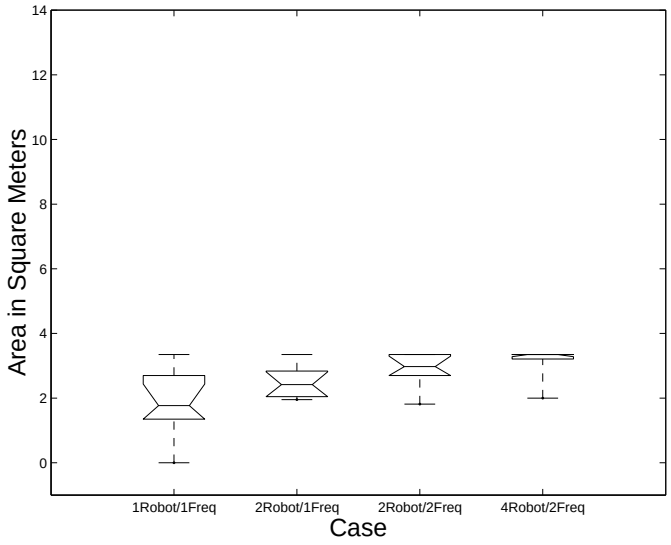


Fig. 16. Experiment C: Detecting Motion. The areas traversed by the Pioneer that the Scouts were able to view.

smaller is the opportunity for the Scout(s) to detect the Pioneer even when there is no frequency swapping. The figures also illustrate the advantages of using a larger number of Scouts. Both the viewable area traversed by the Pioneer and the time that the Pioneer was in view have higher means and smaller variances when more Scouts were used. This provides a justification for the use of more Scouts than strictly needed to cover the area. Given the chance the Scouts will not hide in good places, using more Scouts reduces the variability in the results and provides more opportunities for the detection of motion.

However, we should caution that the differences were not always statistically significant at the $95\,\%$ confidence level. In particular, four robots were found to be significantly better than

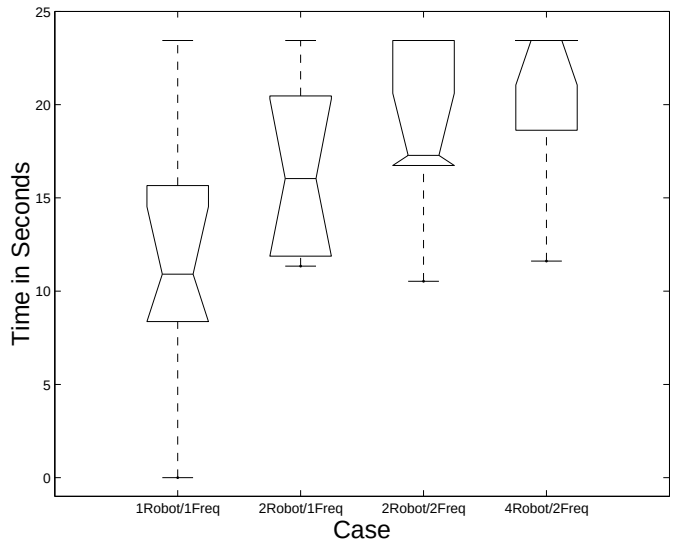


Fig. 17. Experiment C: Detecting Motion. The potential time that the Scouts could have been able to view the Pioneer. This is calculated as the amount of time the Pioneer was in the field of view of a Scout even if the Scout was not active at the time.

one in these measures, but four robots were not found to be significantly better than two on different frequencies for either measure, and two robots on the same frequency were not found to be significantly better than one for Pioneer path area viewed. This is due to the overall better placement of two Scouts using two different frequencies than two Scouts on the same frequency. If the two robot results (cases 2 and 3) are pooled to give a larger sample size, then two Scouts are significantly better on these measures than one, and four are significantly better than two. Pooling these results is justified, as the differences between their means are not significantly different, but we cannot rule out the slight possibility that these results are real effects of the differences in robot interactions in these two cases, rather than simple random noise.

VI. Analysis

When deploying a group of Scouts to create a sensor net, we need to be able to predict their success at detecting motion. Ideally, we would like to guarantee that any motion in the environment will be detected. This clearly depends on the number of sensors in the network, their placement, the communications bandwidth, and the size of the area covered.

There is a tradeoff between placing a large number of Scouts and being able to process their visual information. Many Scouts can view a potentially larger area and provide for redundancy in case of failures. However, increasing the number of Scouts increases the load on the communications channels. When Scouts share video channels, the effectiveness of their detection abilities decreases. Consequently, the number of available video channels is the major factor which limits the number of Scouts that can be used effectively.

The motion detection problem we have presented is similar to the Art Gallery problem [9], [10], in which a robot attempts to find a minimal number of observation points allowing it to survey a complex environment. Our problem is complicated by the fact that the Scouts have a limited field of view, and that incidence and range constraints significantly affect their ability to detect motion. In [11], a randomized algorithm for sensor placement is proposed, which takes incidence and range constraints into account, but not the field of view.

More importantly, we are interested in detecting motion, not just in covering an area. As we will show, the peculiarities of our motion detection algorithm combined with the limited field of view of the Scouts make detection of motion much more complicated. In addition, we are not free to place the Scouts in their best viewing position—they have to find a hiding place autonomously. Finally, since Scouts cannot place themselves in open areas, where they are likely to be seen or stepped on, the size of the environments they can cover is limited by the maximum distance at which they can detect motion.

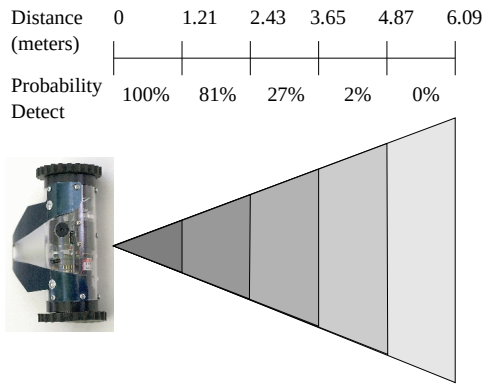
We are interested in using our extensive experimental results to analyze the factors that affect the probability that motion will be detected, and how they affect it. Factors we have considered are: (1) distance, background, and direction of motion which affects the motion detection algorithm and (2) size and shape of the environment which affects the placement of the Scouts.

We have not considered other factors that could affect performance, such as taking into account explicit knowledge of the motion of the moving object(s). Even though in our experiments we have used a single object moving at constant speed on a straight line, we do not use any of this information in the motion detection algorithm.

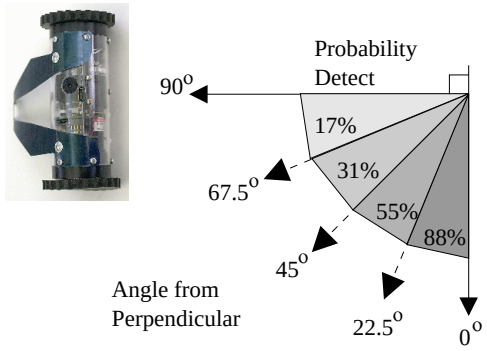
A. Factors Affecting the Motion Detection Algorithm

Our motion detection algorithm (described earlier in Section IV) works by computing the difference between sequential frames of video. The algorithm fails if the motion is not large enough to be distinguishable from RF noise. When the target is too far away from the camera, the motion between subsequent video frames is too small to be detected. Figure 18(a) shows how the probability of detecting motion decreases as the target distance increases. Additionally, when the target moves almost parallel to the optical axis of the camera, then there is not enough difference between subsequent video frames to detect motion. Figure 18(b) shows how the probability of detecting motion changes with the direction of the movement of the target with respect to the Scout. An additional factor that affects the ability to detect motion is the background. If the target is the same color (or intensity for grayscale video) as the background, the motion detection algorithm will fail to detect anything. We ignore this factor in our calculations, since we run our experiments in an environment full of clutter where the target is unlikely to blend into the background for much of its motion.

The experimental evidence we collected on the effect of distance and incidence in detecting motion of an object as large as a Pioneer 1 shows that the Scout cannot be further than $4.87\,\mathrm{m}$ from the moving object. Combining this with the fact that the Scout's video camera has a field of view of $48\,^{\circ}$ makes the maximum area that one Scout can theoretically monitor $9.95\,\mathrm{m}^2$.



(a) The further the moving object is from the Scout, the less likely it will be detected.



(b) The closer the direction of motion comes to being parallel to the optical axis, the less likely the Scout will detect it.

Fig. 18. How the target's distance from the Scout and the direction of the target's motion affects how well the Scout can detect it.

B. Factors Affecting the Placement of the Scouts

To detect motion, Scouts must be placed in areas of open space through which targets are likely to move. These spaces should not be longer than the range in which the Scouts can effectively track motion. Our problem is complicated by the fact that Scouts have to autonomously find their hiding places and they cannot be placed precisely to minimize the required number, as in the Art Gallery problem. In addition, Scouts have to hide to avoid being seen or stepped on. Scouts tend to hide on the periphery of the open area facing toward it. Because of this, the best type of environment for them is a convex one which is no larger than approximately 5 m across. Since any motion will happen in the central open area, the Scouts place their backsides (their blind spots) next to the walls where no motion can take place. Large complex environments can be subdivided into smaller regions. Figure 19 illustrates such a subdivision. For full coverage, each convex region needs its own set of Scouts.

C. Paths of Motion

A priori knowledge about the motions expected in an area can help in determining the number of Scouts needed, the sharing

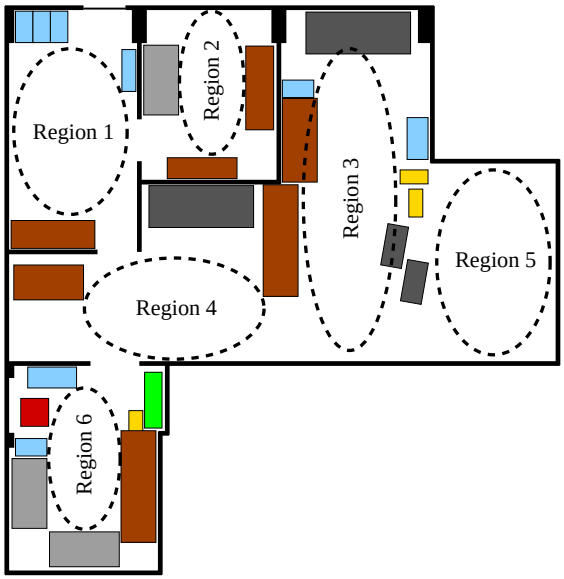


Fig. 19. A top-down view of a complex (multi-room) environment and how it could be broken into multiple smaller convex regions. Each region would have its own Scout (or set of Scouts) to monitor it.

of the bandwidth, and the choice of the motion detection algorithm. For instance, assuming there is a single moving target, Lavalle [12] proposed strategies for maintaining the visibility of the moving target with a moving observer. Pursuit-evasion has been studied as a computational geometry problem. Guibas *et al.* [13] provide bounds on the number of pursuers needed to track an evader depending on the geometric and topological complexity of the environment.

We are interested in a more general setting, where there are multiple observers, each with limited motions, limited computing power, limited communications channels, and potentially multiple targets.

In the results reported here, we use knowledge about where the motion occurs only to measure how well the Scouts do the task and extrapolate from our experimental results how well we should expect them to do in a different environment. In all our experiments motion occurred, and the Scouts detected it 92 % of the time. However, this does not help us understand what factors affect the performance. We know that motion occurs only on a path through the center of the region covered by the Scouts. We assume that every cell within that path has detectable motion at some point during the time that the Scout is observing it. This reduces the problem to determining how much of the Scout's field of view intersects with the path taken by the target and computing the probability that motion will be detected in those cells.

D. Probabilistic Model of Motion Detection

We make two assumptions. First, the size of the environment is known. If this is not the case, then exploration must be done to acquire the missing information. Second, a detectable motion could occur in any location and at any time. We discretize the space using a grid and assume that motion will occur with the same probability in any cell.

We will use the following binary random variables:

det_{ij} = the i-th robot detects motion in the j-th cell,
c_i = the camera of the i-th robot is on,
m_j = motion occurs in the j-th cell

We know that $P(det_{ij} \wedge \neg c_i) = 0$ since nothing can be detected by a robot when the camera is not on. So, we have

$$P(det_{ij}|m_j) = P(det_{ij}|m_j \wedge c_i)P(c_i)$$

$P(det_{ij}|m_j \wedge c_i)$ is the probability the i-th robot detects motion in the j-th cell, given that motion occurs in the j-th cell and the camera of the i-th robot is on. s_{ij} is defined as:

$$s_{ij} = P(det_{ij}|m_j \wedge c_i).$$

This is the quantity we measured in our experiments.

When multiple robots are used there are two complicating factors: (1) their cameras might share the same communications channel, (2) their fields of view might partially overlap. We are primarily interested in the probability that motion is seen by at least one robot. In the case of two robots, assuming they both see the same j-th cell, this is expressed as follows:

$$\begin{aligned}
&P(det_{1j} \vee det_{2j}|m_j) \\
&= P(det_{1j}|m_j) + P(det_{2j}|m_j) - P(det_{1j} \wedge det_{2j}|m_j) \\
&= s_{1j}P(c_1) + s_{2j}P(c_2) \\
&\quad - P(det_{1j} \wedge det_{2j}|m_j \wedge c_1 \wedge c_2) \cdot P(c_1 \wedge c_2) \\
&= s_{1j}P(c_1) + s_{2j}P(c_2) \\
&\quad - P(det_{1j}|m_j \wedge c_1) \cdot P(det_{2j}|m_j \wedge c_2) \cdot P(c_1 \wedge c_2) \\
&= s_{1j}P(c_1) + s_{2j}P(c_2) - s_{1j}s_{2j}P(c_1 \wedge c_2)
\end{aligned}$$

If the two robots share a single channel then $P(c_1 \wedge c_2) = 0$. If the robots are on two different channels then $P(c_1 \wedge c_2) = P(c_1) = P(c_2)$.

In general, given n robots, looking at all possible combinations of detection and camera availability can be prohibitively expensive. In our system, all video channels change from one Scout to the next at exactly the same time. Thus, determining which Scouts are simultaneously active reduces significantly the number of combinations to be considered. For instance, if four Scouts share one channel and six Scouts share a second channel, we can see in Figure 20 which Scouts on each channel are active at any given time. There are only 12 pairs of robots that can have their cameras active at any time.

E. Comparing the Analysis to the Empirical Results

There are two difficulties that arise when trying to predict the performance of the Scouts in an environment. First, the performance of the behaviors which place the Scouts in the environment is highly dependent on the local structure of the environment and is difficult to properly generalize. Secondly, because the placements of the Scouts are difficult to generalize, the overlaps between the video channels on the Scout robots are also difficult to generalize.

We want to answer the question *"How many video channels and robots are needed to detect motion in an environment with a given level of confidence?"* If we assume that the environments are no larger than the ones in which we ran our experiments, then from our results, it would appear that two robots using two video channels would probably suffice since this configuration has a higher mean detection motion time than any of the other cases, as shown in Figure 13. However, since nothing is known *a priori* about the nature of the moving object, four robots will see more of the environment than two robots and have a better chance of detecting motion because they will be more likely to see it from a range of different angles and different distances.

We can assign a value to $s_{ij} = P(det_{ij}|m_j \wedge c_i)$ by integrating over the distances and angles that the Scout saw motion. Given the data shown in Figures 18(a) and 18(b), s_{ij} =38 %.

So, given the four experimental cases, the probability that they will detect motion in exactly one square, $P(det_{1j} \vee \ldots \vee det_{ij}|m_j)$, is given as the following:

Case 1: One robot, one video channel. A single robot has access to 100 % of the bandwidth of the channel and so $P(c_1)$ is 1.

$$P(det_{1j}|m_j) = s_{1j}P(c_1) = 0.38$$

Case 2: Two robots, one video channel. In the experiments, because the camera required a few seconds of warm-up time before the image resolved, $P(c_i)$ was actually only 0.25. The value of $P(c_1 \wedge c_2)$ is 0 because the two cameras cannot be active at the same time.

$$\begin{aligned}
&P(det_{1j} \vee det_{2j}|m_j) = \\
&s_{1j}P(c_1) + s_{2j}P(c_2) - s_{1j}s_{2j}P(c_1 \wedge c_2) = 0.19
\end{aligned}$$

Case 3: Two robots, two video channels. Each robot had access to 100 % of its own bandwidth, so like Case 1, $P(c_i) = 1$ for $i = 1, 2$. Additionally, since the cameras are independent, $P(c_1 \wedge c_2) = 1$ as well.

$$\begin{aligned}
&P(det_{1j} \vee det_{2j}|m_j) = \\
&s_{1j}P(c_1) + s_{2j}P(c_2) - s_{1j}s_{2j}P(c_1 \wedge c_2) = 0.62
\end{aligned}$$

Case 4: Four robots, two video channels. Each robot had to share access to its video channel, so like Case 2, $P(c_i) = 0.25$ for $i = 1, \ldots, 4$. Because there are only two video channels, only two robots will be actively viewing at any time. The schedule is deterministic, similar to what is shown in Figure 20, and so it is known which Scouts are active at any time. For the sake of this example, we assume without loss of generality that robots 1 & 3 are active when robots 2 & 4 are not and vice versa. This pruning allows us to remove terms which are 0, greatly reducing the number of terms. In this case, $P(c_i \wedge c_j) = 0.25$ for $i = 1, 2$ and $j = 3, 4$ since both cameras are on only 25 % of the time as in Case 2.

$$\begin{aligned}
&P(det_{1j} \vee det_{2j} \vee det_{3j} \vee det_{4j}|m_j) = \\
&s_{1j}P(c_1) + s_{3j}P(c_3) - s_{1j}s_{3j}P(c_1 \wedge c_3) + \\
&s_{2j}P(c_2) + s_{4j}P(c_4) - s_{2j}s_{4j}P(c_2 \wedge c_4) = 0.31
\end{aligned}$$

This model suggests that if the number of robots is doubled but the number of video frequencies stays the

Time:	1	2	3	4	5	6	7	8	9	10	11	12
Channel1:	cam_0	cam_1	cam_2	cam_3	cam_0	cam_1	cam_2	cam_3	cam_0	cam_1	cam_2	cam_3
Channel2:	cam_4	cam_5	cam_6	cam_7	cam_8	cam_9	cam_4	cam_5	cam_6	cam_7	cam_8	cam_9

Fig. 20. Multiple robot sharing two video channels. Four robots (0-3) share video channel 1 and six robots (4-9) share video channel 2. This chart shows a typical round-robin schedule of which two robots are active at each time index. The cycle repeats itself after time index 12.

same, the performance of the team to detect motion in a single location in the environment will be halved. Watching an area with two Scouts may have a $61\,\%$ chance to detect motion, but if the Scouts are not looking where the motion occurs, they will not detect anything. To decide how many Scouts to use, the size of the environment needs to be taken into account. If it is likely that a small number of Scouts can cover most of the area, then fewer robots (preferably with different video channels) are desirable. However, if the environment is very large, so that the percentage of the area covered by the Scouts is much smaller, then multiple Scouts would be preferred. This would be the case even if the individual chances for detecting motion might be less. Formally, this is represented as $P(det_{1j} \vee \ldots \vee det_{ij}|m_j) \cdot area$, where $area$ is the percentage of the area that the Scouts are able to see with their cameras.

As shown in Figure 21, as the size of the environment increases, the probability of detecting motion in each of the four cases decreases. An interesting effect is seen when comparing the 1 Robot/1 Freq case with the 4 Robot/2 Freq case. When the environment size approaches $6.27\,\mathrm{m}^2$, the benefits of having multiple robots, even those that are sharing channels, becomes evident. The 4 Robots/2 Freq case has a higher probability of seeing the target primarily because of the additional area that they can see. The plateaus in the graph represent cases where the Scouts can see the entire area. In this case, the probability of detecting the target is just $P(det_{1j} \vee \ldots \vee det_{ij}|m_j)$ because $area = 1.0$.

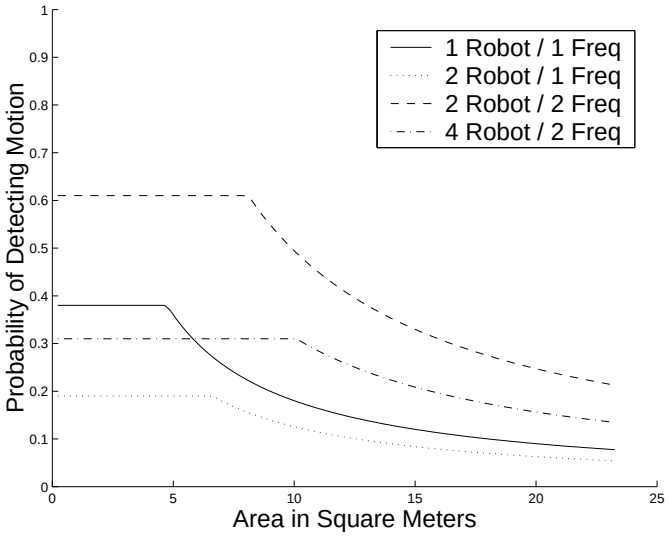


Fig. 21. The average probability of detecting a moving target as a function of the room size and the four different experimental cases.

VII. Related Work

Automatic security and surveillance systems using cameras and other sensors are becoming more common. These typically use sensors in fixed locations, either connected ad hoc or, increasingly, through the shared communications lines of "intelligent buildings" [14] or by wireless communications in "sensor networks" [15], [16], [17]. These may be portable to allow for rapid deployment [18] but still require human intervention to reposition when necessary. This shortcoming is exacerbated in cases in which the surveillance team does not have full control of the area to be investigated. Our system is designed to require as little human intervention as possible. The Scouts have the ability to reposition themselves if they initially place themselves in a bad location. Static sensors have another disadvantage—they do not provide adaptability to changes in the environment or in the task. In case of poor data quality, for instance, we could have our robots move.

Mobile robots such as the Scouts can overcome the problems with static sensors by giving the sensor wheels and autonomy. Robotics research for security applications has traditionally focused on single, large, independent robots designed to replace a single human security guard as he makes his rounds [19]. Such systems are now available commercially and are in place, for example, in factory, warehouse, and hospital settings [20]. However, the single mobile agent is unable to observe many places at once—one of the reasons why security systems were developed.

Because of their small size and portability, many Scouts can be carried into an area for deployment by a human or another robot. Multiple Scouts can simultaneously monitor a much larger area than a single robot could. Further, mobile robots larger than the Scouts are unable to conceal themselves, which they may need to do in hostile or covert operations. They may also be too large to explore tight areas. These are environments which the small size of the Scout robots gives them an advantage over a single larger robot. Multiple mobile robots for security have recently been investigated [21]. In this case, the robots were meant to augment human security guards and fixed sensor systems in a known and semi-tailored environment. In the task we describe in this paper, the Scouts are fully autonomous.

Recently there has been a significant interest in miniature robots. Constructing robots that are small, easily deployable, and yet can do useful work and operate reliably over long period of times has proven to be very difficult. Many problems suggest the use of miniature robots [22]. Most miniature robots have wheels [23], [24], others roll [25].

Energy consumption is a major problem [17] for small

robots, as well as sensors used in sensor networks. Due to their small size and limited power, most miniature robots have to use proxy processing, as in Inaba *et al.* [26], 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 Scout robots. Because of their limited size, not only is all processing for the Scout done off-board but also the RF communications is done using only a few channels. This limits severely the ability to control multiple robots at once.

Our software architecture provides support for distribution of resources across robots, use of shared resources, and seamless integration of autonomous and human-supervised control [2]. We need to be able to write missions for teams of heterogeneous robots, as well as handle resource allocation for miniature robots. Other architectures based on components, such as the one described in [27], are meant for small devices with more limited and well defined tasks.

A number of architectures have been developed for robots, many of them described in [28]. Our architecture has some similarities with CAMPOUT [29], a distributed hybrid-architecture based on behaviors. The major difference is that we focus on resource allocation and dynamic scheduling, while CAMPOUT is mostly designed for behavior fusion. We rely on CORBA [30] as the underlying technology for distributed processing, while in CAMPOUT each robot runs an instance of the architecture and uses sockets for communications with other robots. Our architecture has some similarities with ALLIANCE [31], which provides distributed control for teams of homogeneous robots. Our system has been designed for teams of heterogeneous robots and does not impose any restrictions on the methods used for robot control (deliberative or reactive).

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 [32], [33], and modifying the plans when requests cannot be satisfied. Our approach is specially suited to unpredictable environments, where resources are 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 [34] and load balancing [35].

VIII. Summary and Future Work

Visual behaviors for simple autonomous operations of a group of Scout robots have been presented. 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 also been shown. Future work is planned to allow the Scouts to use 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 have also presented some important system issues related to the control of multiple robots over a low bandwidth communications channel. We have described a distributed software control architecture designed to address these issues. An essential feature of the architecture is the ability to dynamically schedule access to physical resources, such as communications channels, radios, etc. that have to be shared by multiple robots.

We have demonstrated how the communications bottleneck affects the overall performance of the robots. We have shown initial results of how our system degrades under increased load. The next step is to add more intelligence into the behaviors which will allow them to dynamically adjust their requested runtimes to react to the situation. Additionally, we are examining other kinds of RF communications hardware to increase the number of video channels. The difficulty lies in the Scout's extremely small size and power supply. 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.

Acknowledgments

Material based upon work supported in part 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 and in part upon work supported by the Doctoral Dissertation Fellowship program at the University of Minnesota. This work has also been supported in part by the Microsoft Corporation.

We would like to thank the reviewers for their extremely thoughtful and valuable comments. By following their suggestions we were able to strengthen the presentation and improve the technical content of this paper.

References

[1] P. E. Rybski, N. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson, "Enlisting rangers and scouts for reconnaissance and surveillance," *IEEE Robotics and Automation Magazine*, vol. 7, no. 4, pp. 14–24, Dec. 2000.

[2] S. A. Stoeter, P. E. Rybski, K. N. Stubbs, C. P. McMillen, M. Gini, D. F. Hougen, and N. Papanikolopoulos, "A robot team for surveillance tasks: Design and architecture," *Robotics and Autonomous Systems*, to appear, 2002.

[3] C. P. McMillen, K. N. Stubbs, P. E. Rybski, S. A. Stoeter, M. Gini, and N. Papanikolopoulos, "Resource scheduling and load balancing in distributed robotic control systems," in *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, Mar. 2002, pp. 223–230.

[4] C. L. Liu and J. W. Layland, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46–61, 1973.

[5] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton, "A proportional share resource allocation algorithm for real-time, time-shared systems," in *Proc. IEEE Real-Time Systems Symposium*, 1996, pp. 288–299.

[6] S. K. Baruah, J. E. Gehrke, C. G. Plaxton, I. Stoica, H. Abdel-Wahab, and K. Jeffay, "Fair on-line scheduling of a dynamic set of tasks on a single resource," *Information Processing Letters*, vol. 64, no. 1, pp. 43–51, Oct. 1997.

[7] P. E. Rybski, S. A. Stoeter, M. D. Erickson, M. Gini, D. F. Hougen, and N. Papanikolopoulos, "A team of robotic agents for surveillance," in *Proc. of the Int'l Conf. on Autonomous Agents*, Barcelona, Spain, June 2000, pp. 9–16.

[8] ActivMedia, Inc., Peterborough, NH, *Pioneer Operation Manual v2*, 1998.

[9] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, Aug. 1987.

[10] V. Chvátal, "A combinatorial theorem in plane geometry," *J. Combin. Th.*, vol. 18, pp. 39–41, 1975.

[11] H. González-Baños and J. Latombe, "A randomized art-gallery algorithm for sensor placement," in *Proc. ACM Symposium on Computational Geometry*, 2001.

[12] S. LaValle, H. González-Baños, C. Becker, and J. Latombe, "Motion strategies for maintaining visibility of a moving target," in *Proc. of the IEEE Int'l Conference on Robotics and Automation*, 1997, pp. 731–736.

[13] L. J. Guibas, J. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry and Applications*, vol. 9, no. 5, pp. 471–494, 1999.

[14] J. Porteous, "Intelligent buildings and their effect on the security industry," in *IEEE Int'l Carnahan Conf. on Security Technology*, Larry D. Sanson, Ed., Sanderstead, Surrey, England, Oct. 1995, pp. 186–188.

[15] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Instrumenting the physical world with pervasive networks," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 59–69, 2002.

[16] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. Lesser, "Distributed sensor network for real time tracking," in *Proc. of the Int'l Conf. on Autonomous Agents*, June 2001.

[17] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, Mar. 2002.

[18] D. Pritchard, R. White, D. Adams, E. Krause, E. Fox, M. Ladd, R. Heintzleman, P. Sprauer, and J. MacEachin, "Test and evaluation of panoramic imaging security sensor for force protection and facility security," in *IEEE Int'l Carnahan Conf. on Security Technology*, Alexandria, VA, Oct. 1998, pp. 190–195, Larry D. Sanson, ed.

[19] T. Kajiwara, J. Yamaguchi, Y. Kanayama, S. Yuta, and J. Iijima, "Development of a mobile robot for security guard," in *Proc. of the 15th Int'l Symp. on Industrial Robots*, Tokyo, Japan, 1985, vol. 1, pp. 271–278.

[20] A. Kochan, "HelpMate to ease hospital delivery and collection tasks, and assist with security," *Industrial Robot*, vol. 24, no. 3, pp. 226–228, 1997.

[21] H. R. Everett and D. W. Gage, "From laboratory to warehouse: Security robots meet the real world," *Int'l Journal of Robotics Research*, vol. 18, no. 7, pp. 760–768, July 1999.

[22] D. Gage, "Minimum-resource distributed navigation and mapping," in *SPIE Mobile Robots XV, (SPIE Proc. Volume 4195)*, Nov. 2000.

[23] G. Caprari, P. Balmer, R. Piguet, and R. Siegwart, "The autonomous micro robot ALICE: A platform for scientific and commercial applications," in *Proc. of 1998 Int'l Symp. on Micromechatronics and Human Science (MHS'98)*, Nov. 1998.

[24] R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis, and P. Khosla, "Heterogeneous teams of modular robots for mapping and exploration," *Autonomous Robots*, vol. 8, no. 3, pp. 293–308, 2000.

[25] B. Chemel, E. Mutschler, and H. Schempf, "Cyclops: Miniature robotic reconnaissance system," in *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1999, pp. 2298–2303.

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

[27] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *ASPLOS 2000*, 2000.

[28] D. Kortenkamp, R. P. Bonasso, and R. Murphy, *Artificial Intelligence and Mobile Robots*, AAAI Press/MIT Press, 1998.

[29] P. Pirjanian, T. Huntsberger, A. Trebi-Ollennu, H. Aghazarian, H. Das, S. Joshi, and P. Schenker, "CAMPOUT: a control architecture for multi-robot planetary outposts," in *Proc. SPIE Conf. Sensor Fusion and Decentralized Control in Robotic Systems III*, Nov. 2000.

[30] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Object Management Group, 1998.

[31] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[32] E. M. Atkins, T. F. Abdelzaher, K. G. Shin, and E. H. Durfee, "Planning and resource allocation for hard real-time, fault-tolerant plan execution," *Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1/2, pp. 57–78, Mar. 2001.

[33] E. H. Durfee, "Distributed continual planning for unmanned ground vehicle teams," *AI Magazine*, vol. 20, no. 4, pp. 55–61, 1999.

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

[35] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *Journal of Parallel Distributed Computing*, vol. 7, no. 2, pp. 279–301, 1989.

PLACE PHOTO HERE

Paul E. Rybski (S'98) received an interdisciplinary B.A in Math/Computer Science with an emphasis in Cognitive Science in 1995 from Lawrence University in Appleton, Wisconsin. He received the M.S. in Computer and Information Sciences in 2000 at the University of Minnesota and is currently pursuing a Ph.D. in Computer Science with a minor in Cognitive Science at the same institution. His research interests include behavior-based control, distributed robotic teams, and robotic navigation/localization. He is a member of the IEEE, ACM and AAAI.

PLACE PHOTO HERE

Sascha A. Stoeter obtained his M.S. in Computer and Information Sciences in 1997 from the University of Minnesota. Before entering the Ph.D. program in Minnesota, he was a research assistant at the Institute for Robotics and Process Control in Braunschweig, Germany. He is a member of the Institute for Electrical and Electronic Engineers and Computer Professionals for Social Responsibility.

PLACE PHOTO HERE

Maria Gini is a Professor at the Department of Computer Science and Engineering of the University of Minnesota. She has received the Continuing Education and Extension Distinguished Teaching Award (1995), the Morse-Alumni Distinguished Teaching Professor of Computer Science (1987), the Outstanding Professor Award (1986 and 1993), the Fullbright-Hays Fellowship (1979), and the NATO Fellowship (1976). She was the Editorial Program Co-Chair, International Conference on Autonomous Agents (Agents' 2000), Barcelona Spain, May 2000. She was also a member of the Advisory Board of IJCAI-99. She is on the editorial board of "Autonomous Robots" and "Integrated Computer-Aided Engineering". Finally, she is member of the Executive Council of the AAAI Special Interest Group on Manufacturing.

PLACE PHOTO HERE

Dean F. Hougen (M'96) received his B.S. in computer science from Iowa State University in 1988 with minors in mathematics and philosophy. He received his Ph.D. from the University of Minnesota in 1998 also in computer science with a graduate minor in cognitive science. After serving as an Assistant Professor in the Department of Computer Science and Engineering and Associate Director of the Center for Distributed Robotics, both at the University of Minnesota, Dr. Hougen moved to the School of Computer Science at the University of Oklahoma, where he has founded the Robotic Intelligence and Machine Learning Laboratory. His research includes distributed heterogeneous robotic systems, learning (reinforcement, connectionist, and memetic) in real robots, and evolutionary computation.

PLACE PHOTO HERE

Nikolaos P. Papanikolopoulos (S'88-M'93-S'01) was born in Piraeus, Greece, in 1964. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1987, the M.S.E.E. in electrical engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1988, and the Ph.D. in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1992. Currently, he is a Professor in the Department of Computer Science and Engineering at the University of Minnesota and Director for the Center for Distributed Robotics. His research interests include robotics, computer vision, sensors for transportation applications, and control. He has authored or coauthored more than 140 journal and conference papers in the above areas (thirty five refereed journal papers). He received the Best Video Award in the 2000 IEEE Int. Conference on Robotics and Automation. He was a McKnight Land-Grant Professor at the University of Minnesota for the period 1995-1997 and has received the NSF Research Initiation and Early Career Development Awards. He was also awarded the Faculty Creativity Award from the University of Minnesota. Finally, he has received grants from DARPA, Sandia National Laboratories, NSF, INEEL, Microsoft, USDOT, MN/DOT, Honeywell, and 3M.