MinDART : A Multi-Robot Search & Retrieval System

Paul E. Rybski, Amy Larson, Heather Metcalf, Devon Skyllingstad, Harini Veeraraghavan and **Maria Gini**
Department of Computer Science and Engineering, University of Minnesota
4-192 EE/CS Building
Minneapolis, MN 55455
{rybski,larson,hmetcalf,dsky,harini,gini}@cs.umn.edu

Abstract

We are interested in studying how environmental and control factors affect the performance of a homogeneous multi-robot team doing a search and retrieval task. We have constructed a group of inexpensive robots called the Minnesota Distributed Autonomous Robot Team (MinDART) which use simple sensors and actuators to complete their tasks. We have upgraded these robots with the CMUCam, an inexpensive camera system that runs a color segmentation algorithm. The camera allows the robots to localize themselves as well as visually recognize other robots. We analyze how the team's performance is affected by target distribution (uniform or clumped), size of the team, and whether search with explicit localization is more beneficial than random search.

Introduction

Cooperating teams of robots have the potential to outperform a single robot attempting an identical task. Increasing task or environmental knowledge may also improve performance, but increased performance comes at a price. In addition to the monetary concerns of building multiple robots, the complexity of the control strategy and the processing overhead can outweigh the benefits. We explore these tradeoffs by comparing single robot versus team performance, as well as examining the benefits of increased intelligence in the form of environmental knowledge.

We propose a task of search and retrieval whereby robots locate, collect, and return targets to a home base. Robots are homogeneous and perform independently with a localized goal of target retrieval. The task is a simplified version of minefield clearing where mines are localized using close-proximity sensors such as magnetometers, or of a search-and-rescue task where robots find and retrieve specific targets such as those dropped by air. The MinDART, targets, and some sample landmarks are shown in Figure 1.

In this paper, we describe the latest editions to the Minnesota Distributed Autonomous Robot Team (MinDART) (Rybski *et al.* 1998; 2002) which include the addition of a CMUCam and a signaling beacon. We have also added a new method of localization which is based on the idea of visual homing. We describe our experiences at the AAAI 2002 Mobile Robot Competition and Exhibition. Finally, we analyze how we would expect these robots to perform when compared to the older revisions.

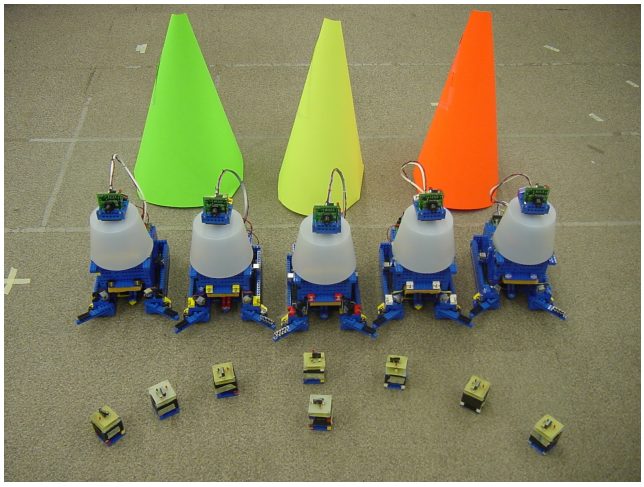


Figure 1: The robots with their infrared targets and colored landmarks.

Related Work

Most research with multiple robots has focused on various forms of collaborative work as detailed, for instance, in (Arkin & Bekey 1997; Cao, Fukunaga, & Kahng 1997). While collaboration may be essential, we are interested in studying tasks that can be done by a single robot, but where using multiple robots can potentially increase performance either by decreasing the time to complete the task or by increasing the reliability. Sample tasks include cleaning up trash, mapping a large area, and placing a distributed sensor network. For this type of task, cooperation usually requires communication among the robots (Earon, Barfoot, & D'Eleuterio 2001; Mataric 1997; Parker 1996). Even simple communication has been shown to substantially increase the performance of robots when foraging, consuming, and grazing (Balch & Arkin 1994). However, direct communication can be replaced by indirect communication via sensing or via the environment (Arkin 1992; Beckers, Holland, & Deneubourg 1994). One example of

this is where a robot uses a beacon to visually transmit data to other robots (Michaud & Yu 1999).

Robotic Hardware

The robots are constructed out of LEGO Technic blocks. LEGOs are used because they are lightweight, easy to work with, and ideal for rapid prototyping. The chassis is a dual-treaded skid-steer design, allowing the robot to turn in place. Each robot is equipped with an articulated cargo bay that is capable of securely grasping a target. For obstacle avoidance, a set of bumpers is located just beyond the front of the robots' treads as well as on the back. Power is provided by two 9.6V NiCad battery packs. Figure 2 shows a close-up of one of the robots.

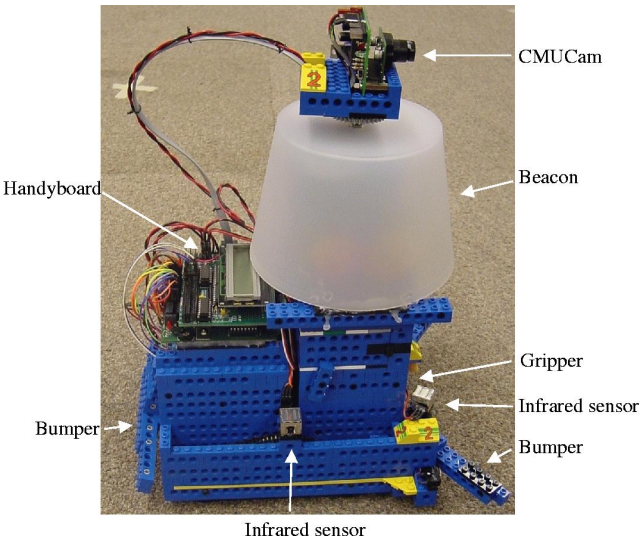


Figure 2: A MinDART robot.

The targets that the robots attempt to locate transmit an omnidirectional stream of $40\,\mathrm{KHz}$ infrared light that is detectable at a range of approximately $70\,\mathrm{cm}$. Two infrared detectors are mounted on each side of the robot and two more are mounted on the front. A CMUCam (Rowe, Rosenberg, & Nourbakhsh 2002) is mounted on top of a servo-controlled turret. The CMUCam is a small CMOS-based digital camera attached to a Scenix SX microprocessor. The Scenix captures frames of video and performs color segmentation on the image. Blob statistics are computed and sent to the robot's on-board computer. A light-bulb beacon sits just under the turret. The robot can activate the beacon to signal other robots. The on-board computer is the Handyboard (Martin 1998), a 2MHz MC68HC11-based microcontroller with 32K of RAM. Because of the Handyboard's slow serial port (9600 bps) and clock speed, it can only receive color information from the CMUCam at approximately 2-3 frames per second. The software was developed in Interactive-C 3.1[1] (Wright, Sargent, & Witty 1996), a subset of C with multitasking capabilities.

[1]We are currently in the process of migrating our code to Interactive-C 4.0.

The CMUCam is a substantial upgrade from the previous sensor configuration. In previous work (Rybski *et al.* 1998; 2002), the robots were equipped only with cadmium sulfide (CdS) photoresistors to detect the presence of light bulb landmarks. Using light bulbs as landmarks was not as flexible because there was no way of distinguishing one landmark from another. With the CMUCam, each landmark has its own unique color. We upgraded our cameras to use a 3.7mm focal length lens rather than the stock 4.9mm lens in order to give them a wider field of view.

In order to accurately determine the bearing to an object, a function which mapped image coordinates to a bearing angle needed to be computed. This was complicated by the fact that the lens had some spherical distortions to it as well as being slightly off-center. The centering problem was an issue with the run of CMUCam boards that we received and has been corrected on later models. To calibrate the camera, a 2D polynomial was fit to a set angular measurements (Bailey *et al.* 1998) . The CMUCam was placed in front of a grid on which a small colored piece of paper was moved to pre-defined positions. The center of mass in the image plane was recorded for each of these positions. The function to fit this data was generated by solving for the coefficients of the following 2D polynomial:

$$\begin{aligned}\theta =& ax^2y^2 + bx^2y + cx^2 + dxy^2 + exy + fx \\ &+ gy^2 + hy + i\end{aligned}$$

Robot software

A set of parallel sensory-motor behavior processes, similar to the Subsumption algorithm (Brooks 1986), are used to control the robot's behavior. Each process is responsible for handling one segment of the robot's control code by mapping sensors to actuators. When a process is activated by a sensor (e.g. when collision detection is activated by a depressed bumper), it tries to control the actuators. In order to resolve conflicts between processes running in parallel, each process is given a unique priority and control of the robot goes to the process with the highest priority.

Finite State Machine Controller

There are several different aspects to the robot's task. They include finding a target, grabbing a target, and returning a target to the home base. Instead of having a single set of behaviors that is responsible for handling all of it, three sets of behaviors were designed. Each set of behaviors corresponds to a specific state in the robot's controller. Depending on the state of the controller, a specific set of parallel behaviors are active. When the controller switches state, the previously-active behaviors are stopped and the behaviors belonging to the new state are activated. Figure 3 shows the finite-state machine control system. On power-up, the control system starts in the *Find Target* state. In this state, the robots search for targets, or head toward a target that they have seen previously. Once a target is detected with the robot's infrared sensors, the control system switches to the *Grab Target* state which is responsible for maneuvering the robot such that the

target fits into the gripper. If the robot successfully grabs the target, the control system switches to the *Return Target* state which will return the robot to the drop-off location. The *Grab Target* state can fail and return to the *Find Target* state either because the robot's infrared sensors lost track of the target for some reason, or because the target was too difficult to grab and the robot gave up. A finite state manager process determines what state the robot should be in and handles the starting and stopping of all the corresponding behavior processes.

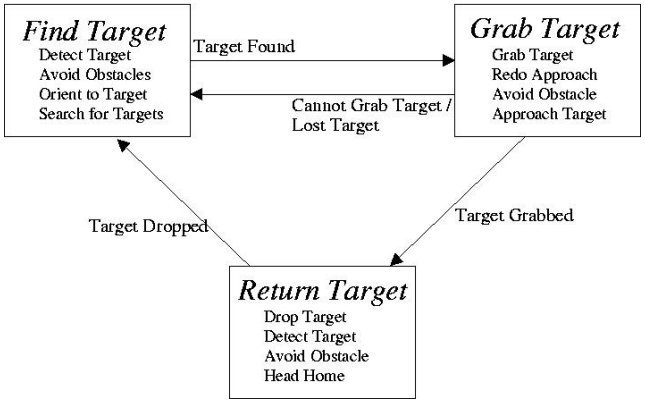


Figure 3: High-level finite-state machine control program.

Behavior Hierarchies

The *Find Target* state has four parallel behaviors in it, each of which attempts to control the robot's motors. The behaviors operate independently of each other and are given access to the motors via an arbiter. As seen in Figure 4, the lowest priority behavior is **Search for Targets** and the highest priority behavior is **Detect Target**. Control of the motors is given to the active behavior with the highest priority.

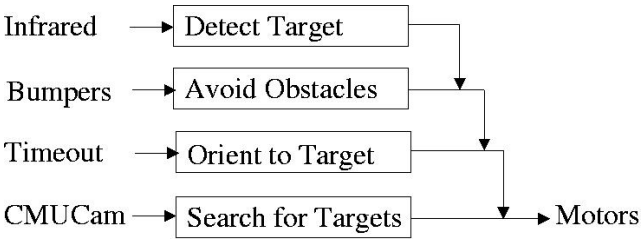


Figure 4: Behaviors in the *Find Target* state.

The **Search for Targets** behavior has the lowest priority and simply drives the robot straight unless the CMUCam sees a robot light up its communications beacon. In this case, the robot servos toward the beacon. Optional parameters to this behavior allow it to change the robot's heading at random intervals.

The **Orient to Target** behavior has third priority and directs the robot to head toward the position of a previously-seen target. If the robot has a position to return to, this behavior activates every 30 seconds, localizes, and rotates the robot to point in the heading of that position. If there is no target visible (possibly because another robot grabbed it already), the behavior will select the next position from the list. If the position list is empty, this behavior will not activate.

The **Avoid Obstacles** behavior has second priority and monitors the five bump sensors (four in front and one in the rear) to detect whether the robot has collided with something. If a collision occurs, this behavior activates and maneuvers the robot back and forth in a circular motion until it is free again.

The **Detect Target** behavior has the highest priority and monitors the infrared sensors for targets. This behavior stops the robot and signals the finite state manager to switch to the *Grab Target* state.

The four behaviors of the *Grab Target* state are shown in Figure 5. This state is responsible for moving the robot so it can grasp the target that was discovered in the previous state.

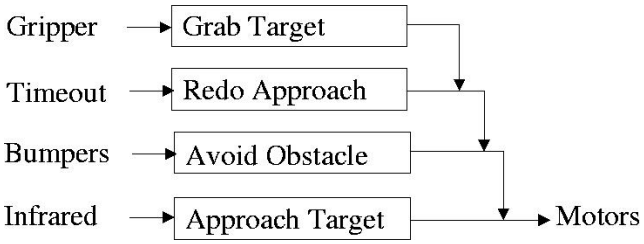


Figure 5: Behaviors in the *Grab Target* state.

The **Approach Target** behavior has the lowest priority and drives the robot toward the target. This behavior rotates the robot until the target is lined up with its front infrared sensors. As long as the target is aligned with the gripper, this behavior will drive the robot straight forward.

The **Avoid Obstacles** behavior has third priority and is slightly different from the behavior of the same name in the *Find Target* state. Unlike the behavior in the previous state, which attempts to move the robot away from all obstacles, this behavior attempts to move the robot so that the obstacle that it collided with is centered on the gripper. This behavior makes the assumption that any collisions are caused by misalignment of the target with the robot's gripper rather than by a stationary obstacle (or another robot).

The **Redo Approach** behavior has second priority and keeps track of how many times the robot has attempted to grab the target. Once a certain number of attempts have been made, this behavior activates and moves the robot to a new position so that it can try to grab the target from a different direction. If this behavior needs to activate more than a few times, it signals the finite state manager that this state has failed and that it should return to the *Find Target* state.

The **Grab Target** behavior has the highest priority and checks to see if a target has moved into the gripper. When this happens, this behavior stops the motors, closes the gripper, and signals the finite state manager to switch the controller to the *Return Target* state.

The *Return Target* state, illustrated in Figure 6, is responsible for directing the robot back to its home base so it can drop off the target it has acquired. This state is also responsible for remembering the locations of other targets that the robot passes on its way home so that it can return to them in the *Find Target* state.

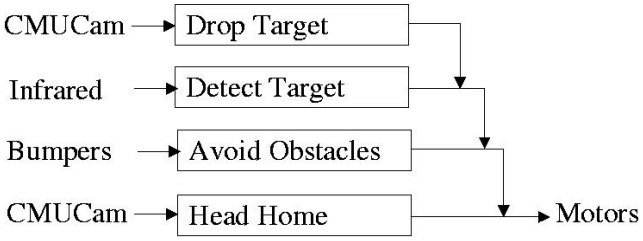


Figure 6: Behaviors in the *Return Target* state.

The **Head Home** behavior has the lowest priority and is responsible for driving the robot back to the home base. There are two different methods for doing this. In one home-base detection method, the robot servos to one of the three landmarks and stops in front of it to drop off the target. This is the simplest method and assumes the robot is capable of accurately determining the relative size of the landmark in the image plane. In the second home base-detection method, the robot stores a specific (x, y) position for its home base and uses its localization routine to determine when it has arrived there. This is more useful when the landmarks that the robot uses are too far away to be able to use their relative sizes as an accurate indication of their distance.

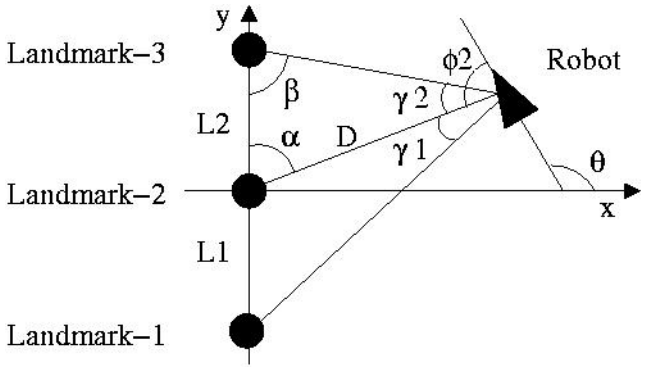
The **Avoid Obstacles** behavior has third priority and is identical to the behavior of the same name in the *Find Target* state.

The **Detect Target** behavior has second priority and monitors the infrared sensors for new targets and stops the robot's motors when one is detected. Since the robot already has a target in its gripper, it cannot grab this new one. Instead, the behavior localizes the robot and stores that location in a list so it can return to it later. If there are already several targets on the stack, the robot does not store the location but rather turns its beacon on in an attempt to recruit other robots to the location.

The **Drop Target** behavior has the highest priority and checks to see whether the robot is at the home base. This is determined either by the size of the home base in the CMUCam's image plane (if the robot is servoing to its home base), or by the current (x, y) location of the robot. This behavior stops the robot, lowers its gripper, and backs the robot away before signaling the finite state manager to return to the *Find Target* state.

Localization

There are two different localization methods available for use by the robots. They are mutually exclusive and have their own advantages and disadvantages. The first localization method assumes that there exist three collinear colored landmarks at known positions in the environment. These landmarks are used to resolve the robot's (x, y, θ) position in a global frame of reference. Figure 7 illustrates the analytical solution to this localization problem.



Let $a = sin(\gamma_1)/L_1$
Let $b = sin(\gamma_2)/L_2$
Let $c = cos(\gamma_1 + \gamma_2)sin(\gamma_1)/L_1$
Let $d = (b - c)/sin(\gamma_1 + \gamma_2)$
$D = \sqrt{1/(a^2 + d^2)}$
$\beta = \pi - (\gamma_1 + \gamma_2) - tan^{-1}(a/d)$
$\alpha = \pi - \beta - \gamma_2$
$x = D * sin(\alpha)$
$y = D * cos(\alpha)$
$\theta = \pi + tan^{-1}(y/x) - \phi_2$

Figure 7: Three-landmark localization used by the robot. The lines connecting the lights and the robot represent the line of sight between the robot and the landmarks. The position of Landmark-2 is the origin of the coordinate system. To avoid problems with symmetry, the robot is only allowed to move in the positive X direction. The values of a, b, c & d are computed through algebraic manipulations involving the Law of Sines and various other trigonometric identities.

The values of L_1 and L_2 are programmed into the robot *a priori* and are assumed never to change. The robot uses its CMUCam to measure the angles to the three landmarks with respect to its own orientation (ϕ_1, ϕ_2, and ϕ_3)[2], thus $\gamma_1 = (\phi_1 - \phi_2)$ and $\gamma_2 = (\phi_2 - \phi_3)$. The angles α and β and the distance to the center landmark D are solved for and from these values, the robot's global pose (x, y, θ) can be calculated. In our approach, the robot's orientation θ is measured with respect to the global x axis.

This localization method typically estimates the robot's position to within 25 cm and its orientation to within 5 degrees. However, it will fail if it cannot resolve three distinct landmarks, such as if a landmark is occluded.

Homing

The previously described localization strategy is useful when the landmarks can be hand-placed in the environment. However, in practice, this is not always desirable. In more general environments, it often more practical to make use of pre-existing features.

[2]For the sake of clarity, only ϕ_2 is shown in the figure.

Visual homing algorithms have been inspired by the navigational strategies of small insects such as bees. Studies suggest that these insects use a combination of the apparent size of the landmarks and the bearings to the landmark for navigation (Hong *et al.* 1991). In essence, they store a snapshot of a location and navigate to that location by minimizing the error between the observed and the stored target snapshots.

Homing can be used by the robots as another method for navigating back to target locations or the home base. The second localization method available to the MinDART is a homing-based algorithm reported by (Weber, Venkatesh, & Srinivasan 1999). A location is stored as a set of bearings to the landmarks from that position. The only requirement of this algorithm is that all of the landmarks must be visible from all parts of the environment. Once the landmarks have been identified, the homing direction is computed. Given a set of landmark bearings, the robot attempts to minimize its landmark error by moving perpendicular to a computed heading so that the current landmark bearings are brought closer to the landmark bearings at the home location. This is illustrated in Figure 8, which shows the path that would be taken by a robot starting from position P_1 to get to the home position. (H_a, H_b, H_c) are the bearings to the landmarks A, B and C as seen from the home position. From a different position P_1, if the bearings to the landmarks are (P_{1a}, P_{1b}, P_{1c}), the correctional vectors are computed such that each vector is in a direction perpendicular to the current bearing and the direction taken brings the bearings of the landmarks closer to those observed from home. The length of the correctional vectors are computed based on the magnitude of the angular difference between the currently observed landmark bearing and the corresponding bearing observed from home. The resultant vector R_1 is just the vector addition of the correctional component vectors. The resultant vector is computed as,

$$R^{\rightarrow} = \sum_{i=1}^{n} V_i^{\rightarrow} \tag{1}$$

where n is the number of landmarks and $V^{\rightarrow}$ are the correctional vectors.

$$V_i^{\rightarrow} = |\theta_i - \beta_i| \angle \delta_i \tag{2}$$

where θ_i is the bearing to a landmark from home position and β_i is the bearing to the same landmark from the current position, and

$$\delta_i = \begin{cases} \beta_i + 90\deg & : \theta_i < \beta_i \\ \beta_i - 90\deg & : \theta_i \geq beta_i \end{cases} \tag{3}$$

Finally, the distance moved by the robot between successive measurements can either be constant or proportional to the magnitude of the total error between the bearings. The second approach seems intuitive in the sense that the larger the error, the longer the distances the robot can move before taking the next measurement; and the smaller the error, the shorter the distance it should move so that it does not overshoot or move past the correct position. However, preliminary experiments indicate that both methods give more or less similar results with our environment, particularly if the robot must make many course corrections due to encountering obstacles and other robots. One of the issues in checking the error between current and home bearings is when a robot approaches a home position from the opposite direction (180 $\deg$ out of phase). One of the ways to deal with this is to use half the total bearing angle computed for moving. Preliminary experiments suggest that both the approaches fare equally well in our setting.

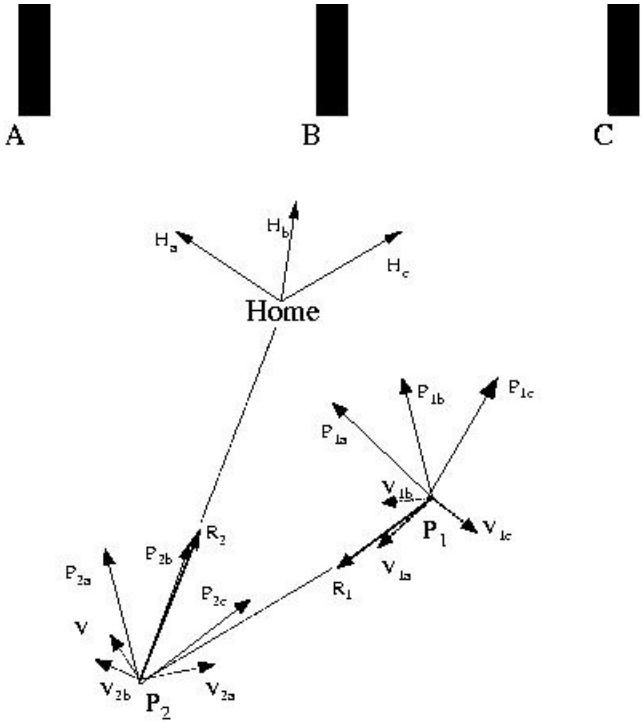


Figure 8: Example of the homing algorithm in operation. The robot is trying to return to the area marked "Home". From position P_1, its first move is away from its intended goal, but from position P_2, its second move is in the correct direction. Such circuitous paths are common when using this algorithm. In this example, the landmarks are all colinear, but they can be placed in any configuration.

Experiences at AAAI

The AAAI 2002 Robot Competition and Exhibition took place in one of the large exhibition halls at the Shaw Conference Center in Edmonton, Alberta, Canada. The large environment in which the robots were allowed to operate is shown in Figure 9. Unlike previous years, there was no enclosed demonstration area. This meant that all demonstrations had to operate without boundaries and had to accommodate spectators walking through the demonstration area. This posed several interesting challenges for the MinDART that we did not anticipate and had to be compensated for onsite. First, the control code was written with the assumption that the robots had a boundary to move around in. Without a boundary, the robots would quickly disperse outside of the

demonstration area. Second, the landmarks that we had prepared for use in our laboratory were too small to be seen at large distances.

Figure 9: MinDART robots performing at the AAAI 2002 Robot Competition and Exhibition. The three cardboard squares against the curtain are colored landmarks used for localization.

In order to properly use all of the space, three $1.11\,\mathrm{m}$ x $0.71\,\mathrm{m}$ landmarks were constructed on-site and set up against a dark curtain on one side of the demonstration area. The landmarks were spaced approximately 5.5 meters apart, ensuring that the robots could see them from nearly anywhere in the demonstration environment. Because the landmarks were square, the robots had to be facing them nearly perpendicularly in order for the cameras to receive the full color. If the robots viewed the landmarks at too oblique an angle, the reflected color was darker than what was detectable by the trained color space. In order to properly view the landmarks, the robots had to stay at least three meters away from them. However, this posed a problem for the *Return Target* state. In our lab, the condition for being home was the relative size and shape of the landmark. The robot could approach the landmark to within $0.5\,\mathrm{m}$ and have the landmark fill nearly all of the visual field. Since the robots couldn't approach the landmarks at the exhibition, the relative sizes couldn't reliably be determined. To fix this problem, the robot's code was modified so that the targets would be dropped off at a pre-defined position in space. Thus, the **Head Home** behavior localized the robot every 30 seconds and when the robot was within a meter of the pre-defined location, it would drop the target.

In order to keep the robots from wandering away from the demonstration area, their code was modified so that they would navigate via way points. If the robots were not actively searching for a previously-seen target, they would follow a path through pre-defined positions. This ensured that they would not go too far away from the demonstration area and would help them cover the space in a more systematic fashion (as compared to random walk).

After these modifications were made, the robot demonstration worked well for most of the conference. An unexpected problem emerged when the environment was suddenly re-arranged to accommodate the poster session on the last day of the conference. The dark green backdrop that the colored landmarks were placed against was removed. This meant that there wasn't enough contrast for the robots to see the targets anymore. The landmarks had to be moved in front of our demonstration booth and the color channels had to be retrained. The only persistent problem was when spectators occluded the landmarks, as seen in Figure 10. When this occurred, the robots could not localize and thus would not navigate properly.

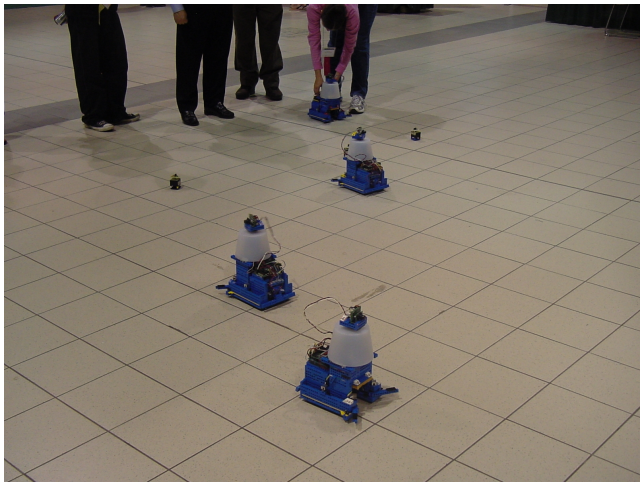


Figure 10: Robot's view of a landmark blocked by spectators.

Even though the robots had a large environment to operate in, they still tended to bunch up around targets and interfere with each other. As seen in Figure 11, three robots are trying to grab the same target and are continually running into each other.

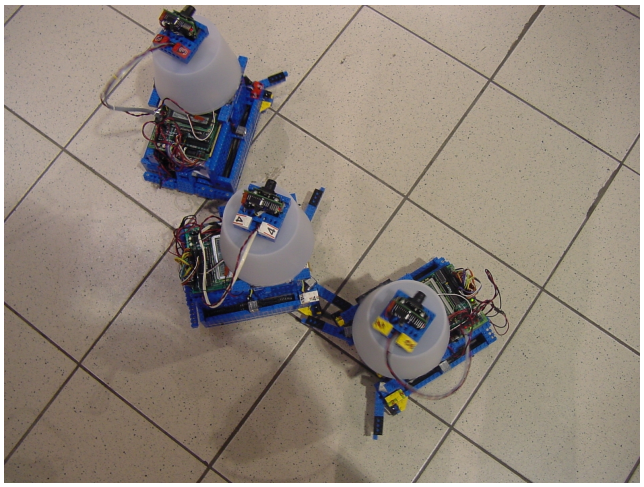


Figure 11: Inter-robot interference.

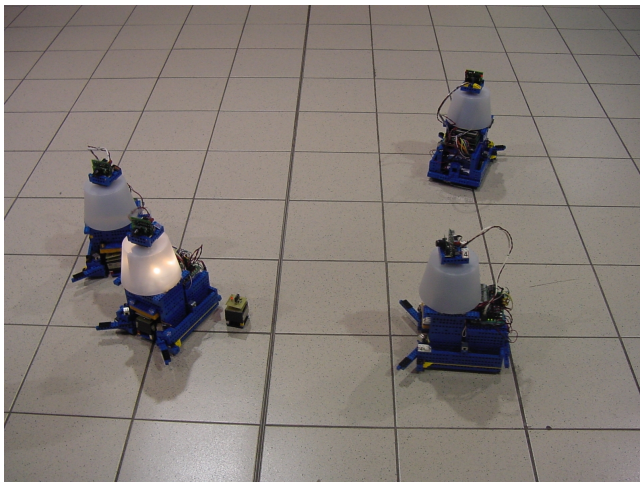


Figure 12: A robot recruits another one to grab a target.

The robots successfully demonstrated their ability to recruit each other as well. The lighting in the Shaw Convention center was such that the beacons could be detected by the robots from about $4-5\,\mathrm{m}$ away. Figure 12 shows a robot (lower right) responding to the lit beacon of another (lower left).

Software and Hardware Challenges

There were several challenges that needed to be overcome when upgrading the robots to use the CMUCam. One problem was caused by interrupt conflicts. The Handyboard controls its servos through the use of an interrupt service routine (ISR). Another ISR was added to the code to monitor the serial port and to buffer characters coming from the CMUCam. This second interrupt would cause occasional timing problems with the first interrupt and so the servos became unstable. This was especially problematic because the camera was mounted on a servo which needed to be stable in order to take accurate measurements to the landmarks. To solve the problem, an external PIC-based servo controller board was added to the robots.

The Subsumption-based robot control architecture works very quickly primarily because reading most of the sensors takes very little time. Each behavior is an Interactive-C process which is written as a series of fast loops. During the execution time of a process, the sensor that the process monitors (bumpers, IR, etc...) is queried many times. Thus there are typically no problems with synchronization if a process is swapped out just after reading a sensor since that sensor will be read multiple times when the process becomes active again. Adding the CMUCam to the robots for the exhibition this year introduced the possibility of priority inversion. Priority inversion occurs when a higher-level process is not allowed to run because a lower level-process is tying up a resource that the higher-level process requires. Reading from the CMUCam and scanning the environment for landmarks is an operation that cannot be interrupted. Doing so would corrupt he bearing readings.

Handling re-entrant code in Interactive-C when actuators are involved is also a problem. Suppose that a low-level process B is controlling the turret servo and is taking readings from the CMUCam. Suppose that B is subsumed by a process A which also wants to use the camera. The state of the servo needs to be saved so that when B's control is restored, it can resume where it left off. If two processes fight over control of the servo and/or reading from the camera, the servo is not physically fast enough to keep up with the conflicting motion commands that are being sent to it. Additionally, if two processes send commands to the CMUCam, the ISR that parses the returned packets will fail to parse the data properly.

Adding mechanisms for avoiding priority inversion and re-entrant code problems added a great deal of complexity to the robot's control code. This kind of complex sensor does not fit very well into our simple Subsumption-based methodology. This suggests that some modifications to this methodology, or at least to the sensor interface, are necessary if such sensors are to be added in the future.

Empirical Performance Analysis

Many factors determine the effectiveness of a cooperative multi-robotic solution to a search and retrieval task. Three such factors include the number of robots used, the physical distribution of the targets, and the kinds of search strategies employed by the robots. The purpose of this work is to study how the overall performance of a robotic team is affected by altering these factors.

Experiments were run where the robots started from a fixed location, searched an area for targets, and returned them to one of three drop-off zones. Experiments were run with one-, two-, and four-robot configurations. Target locations were either distributed uniformly or all lumped together into a single location (non-uniformly). Some experiments were run using localization while others were not. Without the ability to localize, the robots only searched randomly. For a more complete description of the experiments, please refer to (Rybski *et al.* 2002).

Experimental Results

For each of the experiments (results shown in Table 1), the times that the robots returned each target to the drop-off zones were recorded and averaged over five runs for each experiment. The experiments were run until all nine targets were retrieved.

In the earlier work, the robots took 18 seconds to localize themselves. This delay had a significant effect on the overall time to complete the tasks, as reflected in the table. To provide a theoretical upper bound to these calculations, the data was modified so that the times to localize were removed from the pick up and drop off times. These results were labeled "Instant Localization." While these results do not accurately reflect how the robots actually operated, they do provide a useful metric to compare the real robot runs against. Instantaneous localization calculations are also useful because they factor out time bottlenecks caused by implementation details. There are two reasons for factoring out

	uniform 1 robot	uniform 2 robots	uniform 4 robots	non-uniform 1 robot	non-uniform 2 robots	non-uniform 4 robots
no localize	934	458	374	1672	1058	587
18-sec localize	986	478	343	1911	1030	593
5-sec localize	1020	478	326	1490	854	483
instant localize	1108	478	323	1328	***794**	***444**

Table 1: Average time in seconds that the last target was returned to home base. The differences between the starred instant localize results and the no localize results in the same column are statistically significant at the 95% confidence level. Italicized localization data sets were synthesized from the recorded data.

localization overhead. This indicates potential payoff for improvement of the localization technique and it can help determine how much overhead the system can afford while still improving task performance. With the CMUCam to detect the landmarks, the localization routine only took 5 seconds. In practice, instantaneous localization would be difficult to achieve but as can be seen, the delay was reduced from 18 seconds in the original implementation to 5 in the current one. Performance can be analyzed for a range of times for localization, providing a maximum for localization overhead (i.e. the longest localization can take while still improving performance). We wished to see how the team's performance might have been affected with 5-second localization and so we created another set of synthetic data by subtracting only 13 seconds from the original localization times and compared these results against the others.

T tests were run to determine the significance of the non-localization vs. localization trials and the non-localization vs. instant localization trials. Only the two- and four-robot trials with the instant localization and non-uniform target distribution were statistically significant at the 95% confidence interval (one-tailed, two-sample t test, p = 0.0482 and p = 0.0291 for the two- and four-robot cases, respectively.) All other localization results (instant or otherwise) were not statistically significant from the non-localization cases.

In all cases, the robots tended to do much better at finding targets when the targets were uniformly distributed throughout the environment. As was expected, the performance of the robots, regardless of whether they localized or not, was about the same in an environment with uniformly distributed targets. Other targets were rarely encountered when returning one to base, thus localization (i.e. storing the location of a found target) was rarely needed. In contrast, localization was used very heavily in the non-uniformly distributed environment. Robots almost always encountered other targets when returning to base. Robots that did not use localization wandered randomly until they found the cache.

The ability to localize did not necessarily improve the robot team's overall performance due to the computational overhead. The instant localization results were computed to analyze how increased knowledge of the robot's location can affect performance without the overhead of our localization implementation. One thing to keep in mind when looking at the results is that in both the localization and non-localization cases, the robots randomly wandered in the same fashion until the cache of targets was first discovered. Thus, the absolute time that the first target is found affects the absolute times for the other targets. Other factors besides localization overhead contributed to the task completion times. For instance, it is more likely that multiple robots will cluster around the targets and interfere with each other.

One benefit of localization which is not obvious from the results is how quickly the robots were able to find a new target once they had dropped one off. Table 2 illustrates the average times that it took each robot to grab a new target after returning a captured one to the base. Once again, the two localization times and the instant localization results were compared against the no localization results for statistical significance. For this data, all three of the the instant localization with non-uniform target distributions were significant (one-tailed, two-sample t test, p = 0.0085, p = 0.0032, and p = 0.0371 for the one-, two- and four-robot cases.) Additionally, the one- and two-robot cases for the 5-second localization tests were significant (one-tailed, two-sample t test, p = 0.0305 and p = 0.0111 for the one- and two-robot cases). All other localization results (instant or otherwise) were not statistically significant from the corresponding non-localization results.

It is apparent that fast localization allows the robots to more quickly return to a cache of targets that had been seen before, at least in the non-uniform target case. When the robots took the full 18 seconds to localize, their performance was statistically no better than random search.

Conclusions and Future Work

We have analyzed how the performance of a robotic team is affected by environmental factors, the number of robots, and the search strategy employed by these robots. We expected that localization would greatly assist the robots in the non-uniformly distributed environment and not so much in the uniformly-distributed environment. The time that it took to localize greatly affected whether this hypothesis was true. In the 18-second localization case, this was not true. With 5-second localization, this was true in some cases. If the time to localize was completely discounted (instant localization), the robots were much faster at finding their way back to a new target once one had been dropped off. Another hypothesis we had was that adding more robots would greatly increase the performance of the team, but continually increasing the number of robots wouldn't be as beneficial. This was proven true in that four robots generally didn't improve the performance over two robots as much as two robots did over one. Additionally, we observed significant interference between the robots when they tried to obtain targets in the

	uniform 1 robot	uniform 2 robots	uniform 4 robots	non-uniform 1 robot	non-uniform 2 robots	non-uniform 4 robots
no localize	83	57	64	150	181	142
18-sec localize	96	65	79	131	143	152
5-sec localize	91	65	74	***100**	***112**	110
instant localize	89	65	72	***88**	***100**	***94**

Table 2: Average times in seconds for a robot to grab a new target right after a captured one has been dropped off. These values are calculated by the number of targets actually returned during the run. The differences between the starred localize results and the no localize results in the same column are statistically significant at the 95% confidence level.Italicized localization data sets were synthesized from the recorded data.

non-uniformly distributed environment, which added further evidence to this claim. These results show that some knowledge about the structure of the environment is very important when choosing a search strategy for a team of robots.

Future work will include experiments to better analyze exactly how much gain there is by adding more robots to the team. Throughout all the experiments, the obstacles were distributed uniformly. Another variation to consider is a maze-like environment where the targets would be enclosed inside of small alcoves. In this case, explicit localization is expected to be extremely important. Path planning may also prove to be beneficial, if not essential, in this kind of environment. Finally, the effects of communication between the robots will be explored. The implicit communications system using the lighted beacons was not used in the experiments. This and other mechanisms will be examined to see how well they affect the overall performance.

Acknowledgments

This work supported by the Doctoral Dissertation and Louise T. Dosdall Fellowships of the University of Minnesota. Our thanks to Daniel Boley for his insights into the localization algorithm and the help that he gave us in computing a closed-form solution. We would also like to thank Joseph Djugash, Ashutosh Jaiswal, Esra Kadioglu, Elaine B. Rybski, and Lorry Strother for their help in collecting data.

References

Arkin, R. C., and Bekey, G. A., eds. 1997. *Robot Colonies*. Kluwer.

Arkin, R. C. 1992. Cooperation without communication: Multi-agent schema based robot navigation. *Journal of Robotic Systems* 9(3):351–364.

Bailey, B.; Reese, J.; Sargent, R.; Witty, C.; and Wright, A. 1998. Robots with a vision: Using the cognachrome vision system. *Circuit Cellar Ink: The Computer Applications Journal* 92:12–19.

Balch, T., and Arkin, R. 1994. Communication in reactive multiagent robotic systems. *Autonomous Robots* 1(1):27–52.

Beckers, R.; Holland, O. E.; and Deneubourg, J. L. 1994. From local actions to global tasks: Stigmergy in collective robotics. In *Artificial Life IV*, 181–189. MIT Press.

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

Cao, Y. U.; Fukunaga, A. S.; and Kahng, A. B. 1997. Cooperative mobile robotics: antecedents and directions. *Autonomous Robots* 4(1):7–27.

Earon, E.; Barfoot, T. D.; and D'Eleuterio, G. 2001. Development of a multiagent robotic system with application to space exploration. In *Advanced Intelligent Mechatronics*.

Hong, J.; Tan, X.; Pinetter, B.; Weiss, R.; and Riseman, E. 1991. Image-based homing. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 620–625.

Martin, F. G. 1998. *The Handy Board Technical Reference*. MIT Media Laboratory, Cambridge, MA.

Mataric, M. J. 1997. Using communication to reduce locality in distributed multi-agent learning. In *Proc. Nat'l Conf. on Artificial Intelligence*.

Michaud, F., and Yu, M. T. 1999. Managing robot autonomy and interactivity using motives and visual communication. In Etzioni, O.; Müller, J. P.; and Bradshaw, J. M., eds., *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, 160–167. Seattle, WA, USA: ACM Press.

Parker, L. E. 1996. On the design of behavior-based multi-robot teams. *Journal of Advanced Robotics* 10(6).

Rowe, A.; Rosenberg, C.; and Nourbakhsh, I. 2002. A low cost embedded color vision system. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*.

Rybski, P. E.; Larson, A.; Lindahl, M.; and Gini, M. 1998. Performance evaluation of multiple robots in a search and retrieval task. In *Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice*, 153–160. Albuquerque, NM: AAAI Press.

Rybski, P. E.; Larson, A.; Schoolcraft, A.; Osentoski, S.; and Gini, M. 2002. Evaluation of control strategies for multi-robot search and retrieval. In Gini, M., ed., *Proceedings of The 7th International Conference on Intelligent Autonomous Systems (IAS-7)*.

Weber, K.; Venkatesh, S.; and Srinivasan, M. 1999. Insect-inspired robotic homing. *Adaptive Behavior* 7(1):65–97.

Wright, A.; Sargent, R.; and Witty, C. 1996. *Interactive C User's Guide*. Newton Research Labs, Cambridge, MA.