Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection

Andrew Howard
Robotics Research Laboratory
Department of Computer Science
University of Southern California
http://robotics.usc.edu/∼/ahoward

Lynne E. Parker
Distributed Intelligence Laboratory
Department of Computer Science
University of Tennessee
http://www.cs.utk.edu/∼parker/

Gaurav S. Sukhatme
Robotics Research Laboratory
Department of Computer Science
University of Southern California
http://robotics.usc.edu/∼gaurav

Abstract

We describe the design and experimental validation of a large heterogeneous mobile robot team built for the DARPA Software for Distributed Robotics (SDR) program. The core challenge for the SDR program was to develop a multi-robot system capable of carrying out a specific mission: to deploy a large number of robots into an unexplored building, map the building interior, detect and track intruders, and transmit all of the above information to a remote operator. To satisfy these requirements, we developed a heterogeneous robot team consisting of approximately 80 robots. We sketch the key technical elements of this team, focusing on the novel aspects, and present selected results from supervised experiments conducted in a 600 m^2 indoor environment.

1 Introduction

The DARPA Software for Distributed Robotics (SDR) program was designed to demonstrate practical autonomy in large mobile robot teams. Specifically, the program required the design and implementation of an autonomous multi-robot system which could explore and map a large indoor environment, deploy a sensor network into that environment, and use this network to track intruders.

To meet this challenge, we have constructed a heterogeneous team of approximately 80 robots. With a team of this size, cost is an important consideration; robots are therefore divided into two distinct classes, with very different numbers, capabilities and costs:

- A small number of highly capable (and expensive) robots, each equipped with a scanning laser range-finder, camera and powerful onboard CPU.
- A large number of relatively simple (inexpensive) robots, each equipped with a microphone, crude camera, and minimal on-board CPU.

The fully assembled team is shown in Figure 1; the more capable robots are based on the ActivMedia Pioneer2DX and RWI ATRV-mini platforms, while the simpler "sensor robots" are based on the ActivMedia AmigoBot. All robots are equipped with 802.11b WiFi, and are networked together using an ad-hoc routing package. A remote console is used for operator feedback.

Using this team, we divide the overall mission into two phases: 1. exploration and mapping, and 2. deployment and detection. In the first phase, the mapping sub-team explores the environment and generates an occupancy grid map. Exploration is coordinated, and mutual observations are used to solve difficult correspondence problems (i.e., the mapping robots are able to detect and identify one other, thereby resolving potentially ambiguous loop closures). In the second phase, the occupancy grid map is used to compute a set of deployment locations, and the simple sensor robots are deployed to these locations using an "assistive" navigation technique. That is, since the sensor robots are not capable of safe navigation (they lack the necessary ranging sensors)

(a) (b)

Figure 1: (a) The heterogeneous robot team, with two classes of robots; the mapper/leader robots are based on the ActivMedia Pioneer2DX and RWI ATRV-mini, the sensor robots are based on the smaller AmigoBot. (b) The mapping sub-team (four robots): each robot carries a unique laser-visual fiducial that can be detected and identified at ranges in excess of 8m.

they are guided into position by the more capable robots. Once deployed, the sensor robots collaborate to form a distributed sensor network that tracks intruders based on their acoustic signature.

This system has been validated in a series of experiments carried out under very rigorous conditions. An independent team selected and prepared the experimental site, specified the experimental conditions and metrics, and supervised the conduct of individual trials (monitoring completion times, operator interventions, code modifications and so on). Access to the site was limited both prior to and during the experiments, such that the majority of the environment was unknown to the human operators.

In the following pages, we describe the key algorithms used for exploration and mapping, and deployment and detection. We present empirical results derived from supervised experiments, and conclude with some practical observations on the difficulties of managing large mobile robot teams.

2 Mapping and Exploration

2.1 Mapping

Our mapping algorithm employs both centralized and decentralized components: each robot uses an on-board incremental simultaneous localization and mapping algorithm (SLAM) to maintain an independent *local pose* estimate; these estimates are transmitted to the remote operator console, where they are combined through a second SLAM algorithm to generate consistent *global pose* estimates for all robots. An occupancy grip map, combining data from all robots, is generated as a side-effect of this latter process, and used for subsequent deployment operations.

For estimating local pose, each robot employs an incremental maximum likelihood filter [18], similar in spirit to that described in [6]. This filter uses data from a scanning laser range finder to correct for drift in the robot's odometric pose estimate. The state vector for the filter has two components: a local pose estimate and a local map. Given a new range scan, the filter is updated as follows:

1. Fit the scan against the local map.

2. Compute the corrected local pose from the scan fit.
3. Add the new scan to the map and subtract an old one.

The local pose estimates generated in this fashion have the same fundamental properties as odometry: the origin of the coordinate system is arbitrary, and estimates drift over time. Importantly, however, the drift rate in the local pose estimate is least an order of magnitude lower than that seen with odometry alone. Figure 2, for example, shows the local and odometric pose estimates for a single robot; for comparison purposes, the robot's global pose estimate (described below) is taken as the "ground-truth". Whereas the odometry estimate quickly diverges, the error in the local pose is less than 0.1 m after 100 m of travel.

Individual robots do not attempt to close loops or merge data from other robots; this is the role of the remote console, which aggregates local pose estimates and laser range scans from each of the mapping robots into a central repository; the overall map is assembled from this data. Our global SLAM algorithm is built around three key technologies: maximum likelihood estimation, manifold representations and loop closure using mutual observation.

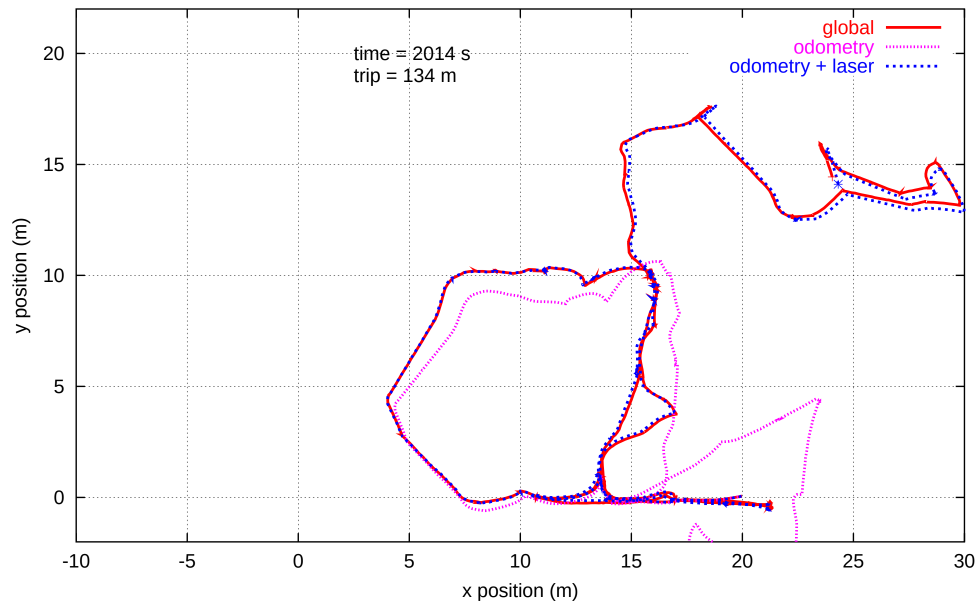
Maximum likelihood estimation (MLE) is used to generate globally consistent maps; put simply, MLE determines the set of robot trajectories that minimizes the global inconsistency between overlapping laser scans [12]. In practice, this is a high-dimensional optimization problem that must be solved using a sparse graph-based representation and numerical optimization. In this context, the use of local pose estimates in the place of raw odometry greatly simplifies the optimization problem: instead of treating each individual laser scan, we assemble successive scans into *local maps* using the local pose estimate, then optimize over the set of such maps. New maps are formed whenever the cumulative uncertainty in the local pose (i.e., since the map was first created) exceeds some pre-set threshold. In these experiments, each local map typically captures between 3 and 10m of robot travel, yielding an optimization problem with a few hundred to a few thousand variables (which is well within the capabilities of contemporary conjugate-gradient optimization algorithms).

We have also made an important extension to the basic MLE formalism: instead of treating the map as a planar structure, we represent it using a two-dimensional *manifold* [7]. Unlike planar representations, this manifold representation is always self-consistent, irrespective of whether or not loops have been closed. To achieve this self-consistency, the manifold representation must sacrifice *uniqueness*; i.e., a single location in the world may be represented more than once in the manifold (see Figure 4). As a consequence, we can reduce loop closure to the problem of identifying and bringing together those points on the manifold that represent the same point in the world.

To recognize such points, we make use of *mutual observations*: if two robots are far apart on the manifold, but are proximal in the world (i.e., they can sense one another directly), we can infer a new set of constraints and close the loop. In practice, each mapping robot is equipped with a scanning laser range-finder, a color camera and a unique coded fiducial, such that they can determine the identity, range and bearing of nearby robots. By exchanging such observations (Figure 3), pairs of robots can determine their full relative pose (i.e., range, bearing and orientation). Each mutual observations is transmitted to the remote operator console, where it is added to the set of global constraints; if the robots lie on two remote points in the manifold, the new constraint will tend to bring those points closer together, thus closing the loop.

This approach to loop closure entirely side-steps the hard correspondence problems associated with conventional SLAM algorithms, particularly those operating in self-similar environments; mutual observations provide constraints that are both reliable and unambiguous. On the other hand, this approach requires at least occasional encounters between robots that have travelled through topologically distinct regions of the environment. In the experiments described in this paper, such encounters are entirely accidental, and arise as a side effect of the randomized exploration strategy described in the next section. Fortunately, the structure and size of the environment is such that suitable encounters are relatively frequent. [1]

[1] There are principled methods for planning robot encounters [16]; application of these methods would form a natural extension to the work described here.



(a)

Figure 2: Comparison of global pose estimates (solid line) with the odometric (dotted) and local (dashed) pose estimates. The robot starts from the bottom right corner of the plot and travels approximately 100 m. The final local pose estimate is within 0.1 m of the global value.

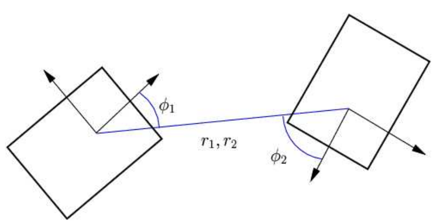


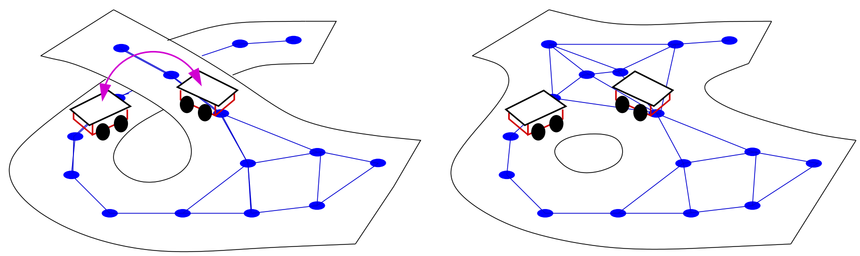
Figure 3: Mutual observation between two robots. Each robot measures the identity, range and bearing of the other robot; by combining pairs of such observations, the relative pose of the robots (range, bearing and orientation) can be determined.

2.2 Exploration

For exploration, we employ a decentralized frontier-based approach [22, 2] with local occupancy grids [5] and minimal communication between robots. The basic algorithm (running on all robots) is as follows.

1. Construct a local occupancy grid map using laser range data and local pose estimates.

This map is "local" in both the spatial and temporal sense: since local pose estimates are subject to drift, one cannot meaningfully fuse observations that are widely separated in either space or time. Therefore, the local map considers nearby regions and recent observations. As an implementation detail, note that the additive properties of occupancy grids are such that this map can be constructed incrementally: new observations are added to the map and simultaneously pushed onto a queue; old observations are popped from the queue and subtracted from the map.



(a) (b)

Figure 4: A manifold map representation, before and after loop closure. The manifold is a self-consistent – but redundant – representation, in which the same point in the world may appear more than once on the manifold. (a) Robots at remote locations in the manifold observe that they are proximal in the world (i.e., the robots detect one another using their sensors). (b) This information is propagated through the manifold, allowing the loop to be closed.

2. Extract a list of discrete *frontiers*, i.e., the boundaries between known and unknown regions of the occupancy grid. Discard frontiers that are unreachable.

This is a multi-step process in which individual frontier cells are first labeled and then grouped into connected components. Unreachable frontiers are detected by constructing the configuration space of the occupancy grid (i.e., expanding occupied cells by the radius of the robot), and applying Dijkstra's algorithm [4]. Figure 5 illustrates these steps.

3. If the currently selected frontier has disappeared or become unreachable (due to obstruction by another robot), randomly select a new frontier.

The nature of frontiers is such that they necessarily recede as the robot approaches; this algorithm selects and pursues a receding frontier until such time as that frontier disappears entirely or is obstructed by another robot. At this point, a new frontier is selected and the process continues. Random selection is used to ensure asymptotic coverage; i.e., given sufficient time, every point in the environment will eventually be explored [21]. The rate at which exploration proceeds, however, is determined by the structure of the environment: open or well-connected environments will be explored much more rapidly than cluttered or minimally-connected environments.

This simple algorithm has a number of attractive features: it is fully decentralized (and thus robust to communications failures), does not require global pose estimates (robust to failures or delays on the remote operator console), and is able to de-conflict the robots' actions without explicit communication (conflicting robots appear in the local map as obstacles, prompting the selection of a new frontier for exploration). Unfortunately, it also suffers from two major drawbacks. First, the temporal locality of the map implies a robot may explore the same location more than once (having "forgotten" that this part of the environment has already been covered). Second, the lack of explicit coordination between robots leads to redundant exploration: any given part of the environment may be explored by more than one robot. Both effects can be seen in the results shown in Figure 6; this plot shows the net area explored by a team of four robots as a function of total distance traveled. Note the rapid initial exploration, followed by slow convergence to full coverage. Also shown on this plot is the effect of varying the team size: a team of four robots provides only marginal improvement over a team of three. This was, perhaps, the most disappointing result observed in the exploration and mapping phase of our experiments: for exploration that is both rapid and complete, some degree of high-level supervision and/or coordination may be necessary.

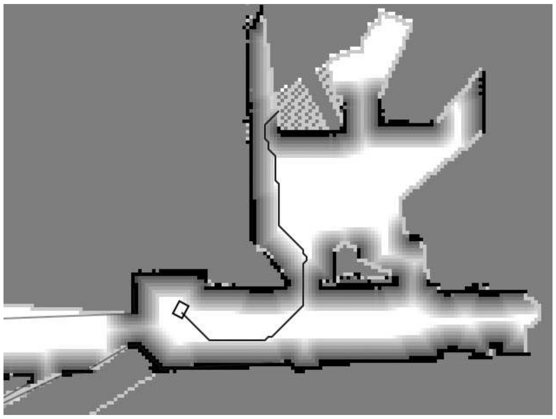


Figure 5: Snapshot of the local map used for exploration. Black cells represent occupied space, white cells are free, dark gray cells are unexplored. Frontiers are marked in light gray, and a frontier is said to be reachable if there exists a collision free path joining it to the robot's current location.

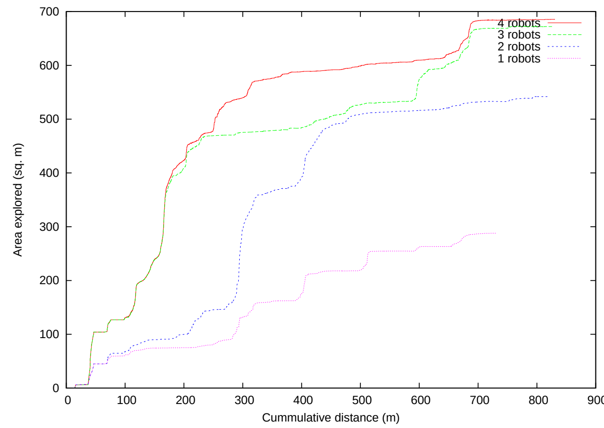


Figure 6: Coverage plot for varying numbers of robots: the plot shows the net area explored by the robots as a function of the total distance traveled.

2.3 Mapping and Exploration Results

Formal experiments were conducted with a team of four robots equipped as shown in Figure 1(b); each robot has a SICK LMS200 scanning laser range-finder, a Sony PTZ camera and a pair of fiducials to facilitate mutual recognition. Maps were built on the remote operator console, using data transmitted over an ad-hoc 802.11b wireless network (total bandwidth for four robots is approximately 3 kB/s, consisting primarily of laser scan data). Since communication in this environment was expected to be patchy (and some robots did indeed experience communication blackouts lasting several minutes), we chose to use a UDP-based communication protocol and accept some degree of data loss. The exploration algorithm is such that any gaps created by lost data tend to be filled-in by other robots.

A total of five exploration and mapping trials were conducted under supervised conditions: in the first three trials, all four robots were deployed from a common starting point; in the later "challenge" trials, robots were deployed from one of two widely separated locations (simulating multiple entry points into the environment). One such trial is illustrated in multimedia extension 1. Figure 7 shows the occupancy grid maps generated in each of the five trials, along with the "ground-truth" floor-plan created with pencil, paper and tape-measure. These maps were generated in real time, with an accuracy comparable to or better than that achieved by the human survey team. The mapping process was entirely autonomous, with the exception of occasional operator interventions to expedite the (very slow) exploration process. The results for the challenge trials should also be noted: in these trials, two robots were deployed from each of two entry points. Since the relative pose of these entry points was unknown, each pair of robots was required to explore and map independently, giving rise to unconnected maps. In both trials, however, a robot from the first pair soon encountered a robot from the second, and, as a result of this mutual observation, the two maps were merged into one.

Since the global mapping algorithm is both centralized and unbounded, processing time scales linearly with the number of robots and super-linearly with the size of the environment. In these particular trials, the mapping algorithm running on the remote operator console was able process data slightly faster than it was generated by the four robots, but this probably represents the limit of what is achievable by the current implementation. The addition of more robots or larger environments would necessitate significant optimizations or algorithmic improvements.

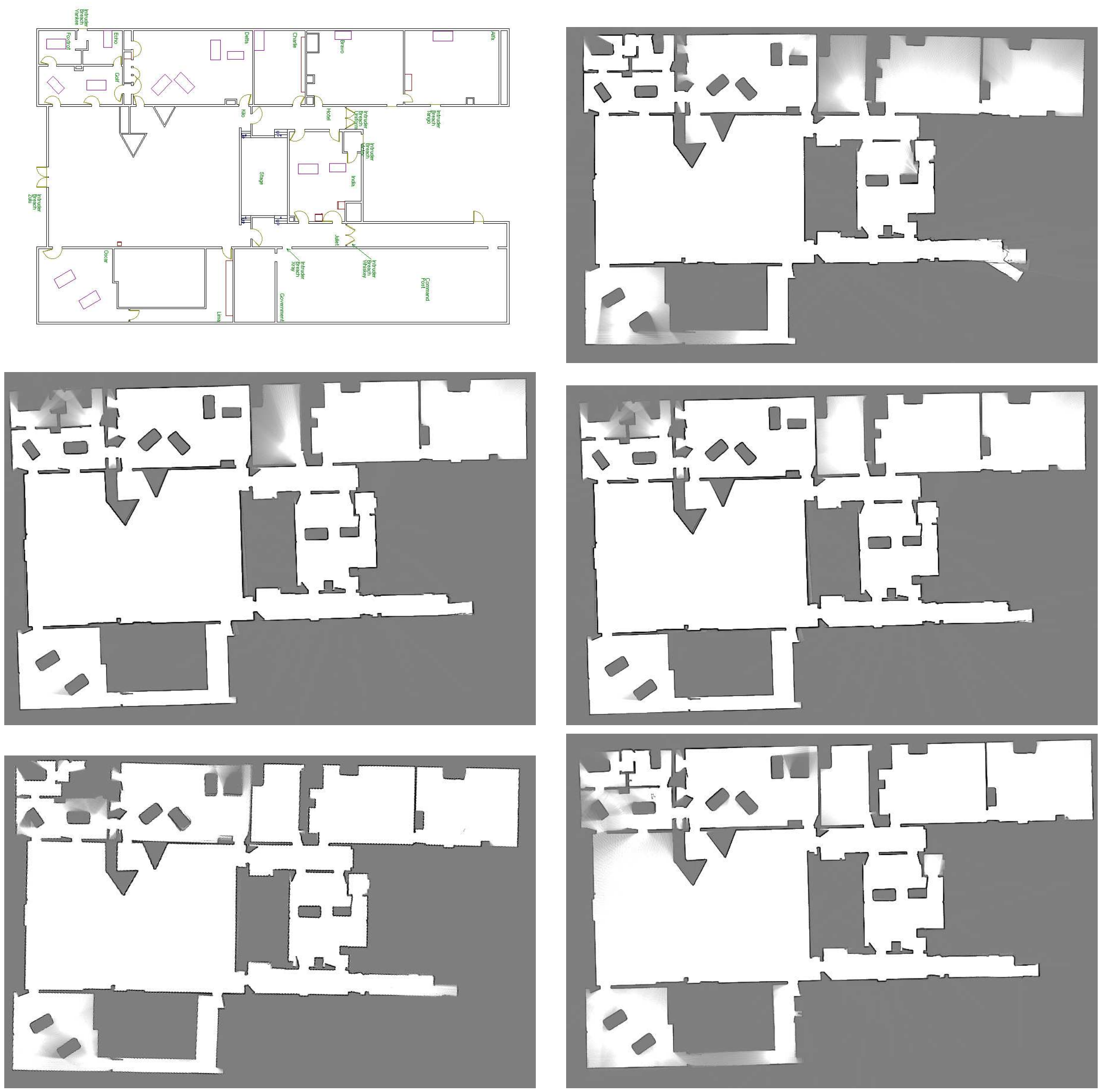


Figure 7: Occupancy grid maps generated for the five supervised experiments. The corresponding floor-plan is shown on the top left; the environment is 45 m by 25 m in size, with an internal area of 600 m^2.

3 Deployment and Detection

The intruder detection and tracking task requires the deployment of a large number (up to 70) of simple sensor robots to serve as a distributed sensor network. The hardware design of these simple sensor robots was driven primarily by cost, project scheduling constraints (18 months from start to finish), and the desire to use commercially-available hardware, rather than custom-build 70 robots. The ActivMedia AmigoBot was chosen as the mobility platform, due to its relative low cost. Each robot was outfitted with an iPAQ PDA running Linux for computation and wireless communication. The iPAQ also included a simple, low-fidelity, non-directional microphone useful for generating the acoustic sensor network. However, obstacle detection and localization sensors were more problematic from a cost perspective, since the proximity sonar sensors commercially available for the AmigoBot were not cost-effective for a team of 70 robots. The custom installation of crude proximity and obstacle avoidance sensors, such as IR or whiskers, were also not cost effective from a technician manpower perspective. The low fidelity of the iPAQ microphones made them impractical for acoustic relative positioning among the robots at a resolution better than 2 meters. Localization based on wireless strength of signal was considered, but was not found to be of sufficient resolution (approximately 0.5 - 1 meters) for independent robot navigation. Thus, the commercially-available CMUCam[2] was selected to enable leader-assisted deployment of the simple robots.

The cost-driven hardware design choice clearly has a significant impact on the software control of these robots. If our simple sensor robots had the ability to detect obstacles and each other (beyond the small field of view of the color tracking camera), then swarm-type approaches to deployment would be appropriate, such as reported in [8, 15, 3]. However, since these abilities are absent on our simple sensor node robots, they cannot navigate safely on their own. Thus, our approach provides cooperative assistive navigation to the sensor robots through the use of more capable leader robots. The leader robots used for the sensor net deployment were three (3) Pioneer 3-DX robots equipped with a forward-pointing SICK laser scanner and a rear-pointing pan-tilt-zoom camera (with resolution 160 by 120 pixels), along with a wireless mobile ad hoc networking capability. Using the map generated in the first phase, these leader robots are able localize themselves and guide the sensor robots to their deployment positions.

The process begins with a pre-planning step that uses the map to determine desired sensor deployment positions that maximize area coverage while maintaining clear pathways for robot deployment. Then, the basic deployment method is as follows: sensor robots are assembled into chains behind a leader using simple color blob tracking (multi-robot follow-the-leader); once a deployment destination is reached, a single robot in the chain is autonomously 'tele-operated' by the leader to the correct position, using the camera mounted on the leader; the leader and the remaining chain then proceed to the next deployment position. Thus, the leader visits a series of locations in turn, and deploys a single sensor robot at each. Figure 8 shows a series of snapshots of our navigational assistance system in operation.

3.1 Planning Sensor Deployment Positions

With this assistive navigation approach, an especially challenging problem is for the leader robots to take paths that the simple robots can easily follow while moving in a follow-the-leader formation. Obviously, the navigational challenges grow if the leader robots move to random sensor node deployment positions without taking into account the formation of robots that is following behind, and the desired deployment positions of the entire group of sensor node robots. Our approach therefore begins with three planning steps. These planning steps occur in a centralized process that runs on the base station. The results of the planning process are then distributed to the leader robots for the actual deployment.

1. Generate the planned sensor deployment positions to meet several criteria, including minimizing pathway obstruction, achieving a minimum distance between sensor robots, and maximizing visibility coverage.

Our approach explores a tree-like structure of potential sensor deployment positions to find locations that satisfy a number of geometric constraints. The first candidate sensor deployment position is generated at a particular location of interest in the map, whose position is supplied to the planning process. The algorithm then generates candidate sensor positions by ray sweeping (at 5-degree increments) from the last

[2]The CMUCam is a low-cost camera that is designed for tracking tasks, such as color blob tracking. The resolution of the CMUCam is 80 by 144 pixels, with a 25 degree field of view. For tracking tasks, the CMUCam returns the position and size of a blob (of a pre-specified color) at 17 frames per second. The frame rate for full image download to the controlling software is considerably slower, since the download occurs over a serial port. Thus, the CMUCam is not practical for applications other than tracking tasks.

Figure 8: Deployment of a sensor robot using assistive navigation: the lead robot first guides and then directs the sensor robot into position.

generated sensor deployment position. Each candidate position must satisfy a number of criteria:

- Within Sensing Range: prefer candidate positions within sensing range of a prior deployment position
- Within Line-of-Sight: prefer candidate positions within line-of-sight of a prior deployment position
- Maximize Visibility Coverage: prefer deployment positions that maximize the new visibility coverage.

Candidate deployment positions are then considered in descending order of the visibility coverage, with preference first going to candidate solutions that meet the line-of-sight criterion. Each candidate position must meet several additional constraints:

- Nearby Obstacle: prefer deployment positions adjacent to obstacles (to reduce pathway occlusion)
- Avoid Doorways: avoid doorways to keep potential deployment pathways open
- Minimum inter-sensor distance: maintain a minimum distance between all deployment positions to ensure sensor coverage
- Room for leader: require room for leader robot to maneuver during deployment.

If the candidate sensor position meets these criteria, it is then added to the list of selected deployment positions. This process iterates until no more new sensor deployment positions can be found. Figure 9 shows results of this planning step. More details of this planning approach are described in [17].

2. Generate the way-points of the path that the leader robot must follow to guide the sensor robots to the vicinity of the sensor deployment positions.

After the sensor deployment positions are planned, the way-points for the leader robot to travel to during the deployment process must be generated, as shown in Figure 10. During the actual deployment process, the leader robot passes through the first planned way-point position and then stops at the second planned way-point position. The leader robot positions are

Figure 9: Autonomously planned sensor net positions (black squares) and planned leader way-points associated with each sensor position (small dots).

planned in such way that the sensor node robot immediately following the leader robot will be adjacent to the planned sensor deployment position when the leader robot stops at the second way-point. In this manner, the sensor node will be properly positioned for deployment by the leader robot using autonomous teleoperation. Figure 9 shows the leader way-point positions corresponding to the previously planned sensor positions.

3. Divide the sensor deployment positions into groups to facilitate the deployment operation.

Since each leader robot can only deploy a few sensor robots at a time, and since several leader robots are available to operate in parallel, each group of positions is assigned to a team (consisting of one leader robot and n sensor robots) for deployment. The physical robot limitations of the simple sensor node robots prevent the sensor nodes from successfully following in a chain through paths that take many twists and turns. In practice, the simple sensor robots in a chain have a strong tendency to get caught on doorways or furniture if the leader's path makes many sharp turns. Therefore, deployment positions must be grouped so

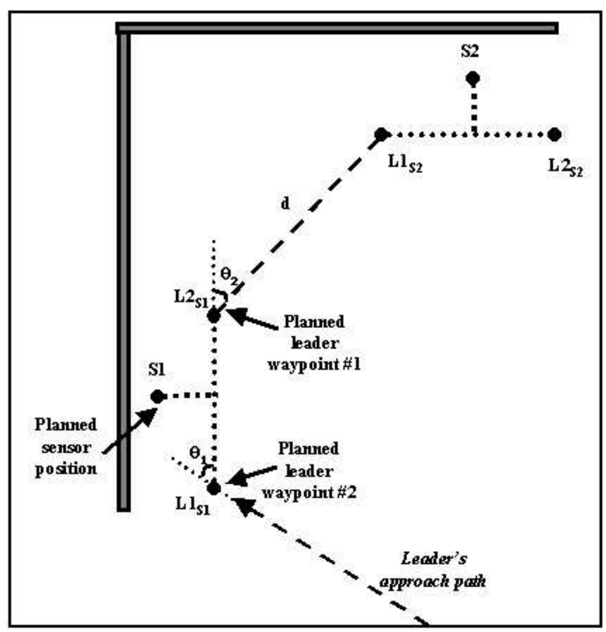


Figure 10: Relationship between a planned sensor position and the two leader positions, and successive sensor positions in a group.

that the path taken to visit each position in the group is as smooth as possible. The team assignments are thus generated to achieve the objectives of: (1) minimizing travel by the deployment teams, and (2) minimizing the amount of turning a team must perform as it travels to all of its assigned deployment positions.

3.2 Deployment

At the beginning of a deployment, the leader robot uses its laser-based Monte Carlo localization capability (similar to [19]) to lead the sensor-limited robots in a chain formation to the vicinity of the goal destination of the first simple robot. (As found by [1], the column chaining formation optimizes formation performance in an obstacle-rich environment). During this navigation mode, the simple robots use a crude camera (the CMUCam) and a color blob tracking algorithm to follow the robot ahead of it, which is outfitted with a rectangular red blob. Each robot in the chain follows the robot directly in front of it at an average deployment speed of approximately 0.5 meters per second. In this mode, the simple robot keeps the red blob within view and moves towards the centroid of the blob. If the blob is lost, the simple robot tries to reacquire the blob by continuing its previous action or by panning itself from side to side. The effect of this blob tracking when multiple robots are front-to-back with each other is a follow-the-leader chaining behavior. This mode of navigation (which we call *Long-Dist-Navigate*) is used when the simple robots are far from their desired destination (i.e., greater than approximately 1 meter).

Once the leader reaches the vicinity of the sensor net position (defined by the way-points illustrated in Figure 10), the leader robot autonomously teleoperates the first simple robot into deployment position (which we call *Short-Dist-Navigate*). To enable the leader robot to determine the position of any simple robot, a visual fiducial (shown in Figure 11) is mounted on each simple robot. This visual fiducial is cylindrical, with a height of 48 cm and a circumference of 23 cm. The marker is composed of four parts: a START block, an ID (identification) block, an Orientation block and an END block. The START and END blocks facilitate the detection of the marker in the field of view. The START block is a combination of red and green stripes at the bottom of the marker. The END block is a red stripe at the top of the marker. Between the START and END blocks are the ID block and the Orientation block. The ID block is unique to each robot, and is composed of 7 black or white stripes, providing $2^7 = 128$ different IDs. The orientation block consists of two stripes that are black for one-half of the fiducial circumference and white for the other half, offset by ninety degrees. Once a marker is recognized in the camera image, the marker detection algorithm determines the identity and the relative position of the marker in terms of the following parameters, as shown in Figure 11:

- d: the distance between the leader's camera and the center of the simple robot
- Γ: simple robot orientation – the angle between the heading of the simple robot and the center of the camera.
- Θ: the angle between the center of the simple robot and the plane containing the leader's camera.

The size and location of the marker in the image enables the leader to determine the location of the simple robot relative to itself, while the orientation block enables the leader to determine the relative orientation of the simple robot about its center axis. Thus, if a marker of height h is located at (x, y) in the image plane of (r, c) pixels, the edges of the marker are (l, r), and the orientation delimitation is located at column k, then the above parameters are calculated by the leader robot as follows:

$$d = \frac{C_1}{h \times C_2}$$

$$\Gamma = 180 \times \frac{k - l}{r - k}$$

$$\Theta = FOV + \frac{x}{c} \times (180 - 2 \times FOV)$$

where FOV is the field-of-view of the camera, and C_1 and C_2 are constants defined by the size of the real marker.

The leader robot combines the relative location of the simple robot with its own global position to determine the global position of the simple robot. The leader robot then communicates steering and control commands to effect the motion of the simple robot

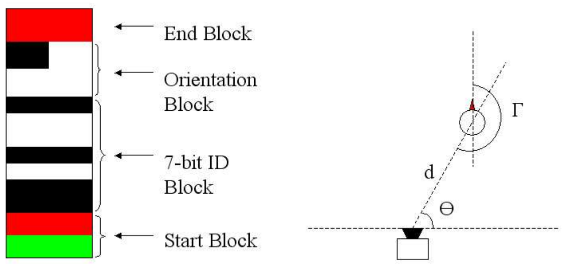


Figure 11: Visual fiducial mounted on sensor node robots to provide unique visual ID, relative position, and orientation information.

towards its global deployment position. To allow for lost communication messages, the simple robot executes the received commands for just a short time period (typically 0.5 to 3 seconds). The leader robot then recalculates the simple robot pose information and sends the next set of control commands, repeating until the simple robot is within a threshold of its planned position.

The behavior organization of the robots to accomplish this deployment process are shown in Figures 12 and 13. In this multi-robot coordination process, several messages are passed between the robots, as defined in Table 1. The simple robots have three main states, as shown in Figure 12: *Long-Dist-Navigate*, *Short-Dist-Navigate*, and *Sensor Net*, in addition to the *Wait* state. The simple robot begins in the *Wait* state until it receives a "Start" message from the leader robot. The simple robot then transitions to the *Long-Dist-Navigation* state, beginning the chain formation-keeping behavior[3]. The simple robots remain in this state until they receive either an "SDN" or "RTW" message from the leader robot, causing them to either switch to the *Short-Dist-Navigate* state or return to the *Wait* state.

In the *Short-Dist-Navigate* state, the simple robot receives navigation control commands from the leader robot to assist the robot in reaching its sensor network deployment position. Once the simple robot reaches its destination position, the leader robot sends an "SNDR" message to the simple robot instructing it to enter the *Sensor Net* state (described further in the next subsection). The simple robot remains in the *Sensor Net* state until a leader robot returns to move the robot to another location. In our application, this occurs when the simple robot's power level falls below a threshold[4] and the robot needs to return to a recharging station. The leader robot becomes aware of this need through messages from the simple robots, and returns to assist the simple robot back to the recharging station.

Figure 13 illustrates the state transitions of the leader robot during the deployment process. The leader robot also has three main states: *Navigate*, *Assist*, and *Transition*, as well as a *Wait* state. Once the leader robot receives a "Start" message (from the human operator), the leader robot enters the *Navigate* state. In this state the leader robot plans a path to the desired deployment location of the first simple robot on its team. It then uses its laser scanner to localize itself and avoid obstacles while it navigates to the goal position. Once the leader robot reaches the goal position, it changes states to the *Assist* state and sends a message to the first simple robot to enter the *Short-Dist-Navigate* state. The leader robot also sends an "RTW" message to the other simple robots on the deployment team to cause them to wait while the first leader robot is being assisted. At this point, the leader robot's goal is to autonomously navigate the first simple robot into its deployment position as described earlier. Once the first simple robot is in position, the leader robot sends it an "ADP" message to let it know that the desired position is reached, followed by an "SNDR" message to cause the simple robot to initiate the sensor net detection mode. Finally, the leader robot sends an "LDN" message to the remaining simple robots, causing them to reinitiate their chaining behavior.

Once the first sensor node robot is guided into its deployment position, the leader robot then successively visits the deployment destinations of the remaining simple robots until all of the robots have been deployed. The leader robot then returns to its home position to pick up another set of simple robots

[3]While an ultimate objective of our research is to enable the robots to autonomously form themselves in the proper physical configuration to enter the *Long-Dist-Navigate* mode, for this research, we assume that the robots are manually initialized to be in the proper front-to-back orientation for successful chaining.

[4]Our simple robots have a battery life of approximately 6-8 hours for this application. Thus, the recharging activity is not a frequent occurrence.

Table 1: Messages defined to achieve inter-robot coordination and cooperation.

Message ID	*Description*	*Sender*	*Receiver*
Start	Start mission	Human op. or leader robot	Leader or simple robots
LDN	Initiate Long-Dist-Navigation mode	Leader robot	Simple robot
SDN	Initiate Short-Dist-Navigation mode	Leader robot	Simple robot
ADP	At Desired Position	Leader robot	First simple robot
SNDR	Initiate Sensor Net Detection mode	Leader robot	First simple robot
RTW	Return to Wait	Leader robot	Simple robot

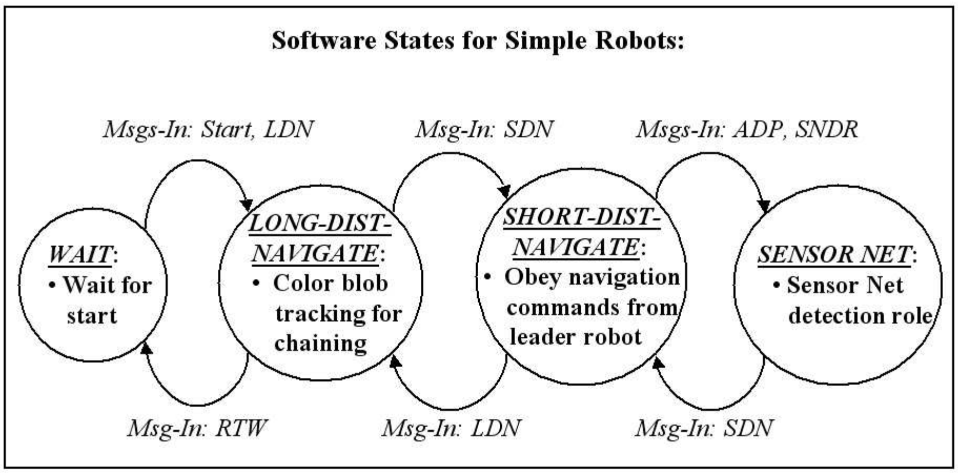


Figure 12: State diagram of simple robot.

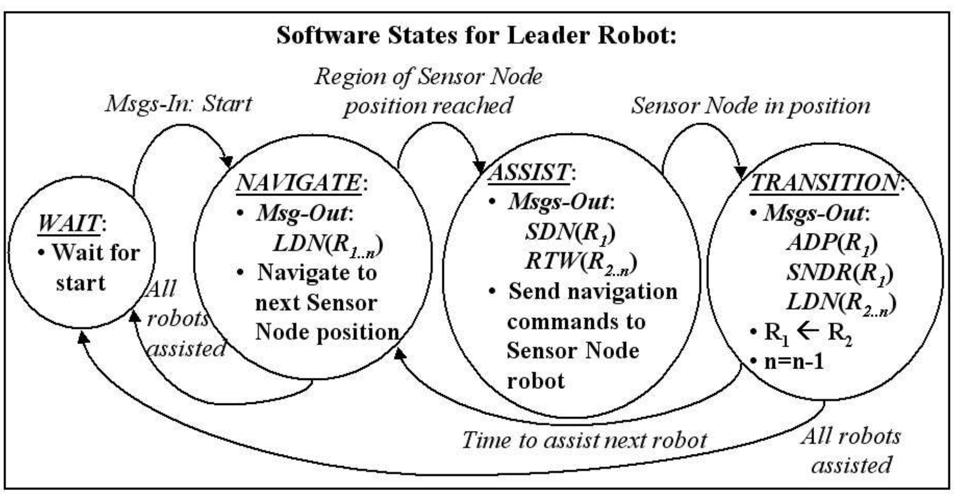


Figure 13: State diagram of the leader robot.

to deploy. This process is repeated until all of the simple robots on the deployment team have reached their desired positions (see also [14]).

3.3 Acoustic Sensor Net Detection

Once the simple robots are in position, they switch state to their primary role of forming a distributed acoustic sensor network for intruder detection. Their objective is to detect acoustic targets that are moving through the environment. We assume that the target is making some detectable noise, and that the target is the only source of sound. Figure 14 shows these robots deployed in the planned sensor net positions and acting as this sensor network.

In our application, each sensor node has only one low-fidelity microphone; these microphones are not calibrated across robots. Individual acoustic sensors are only able to detect volume; the distance and direction to the target cannot be detected. Our real world constraints prevent us from using a known energy output level from the target, both because of practical issues of varying sound levels at the source and declining microphone sensitivity as battery power declines. Sophisticated algorithms for target localization under these constraints have been developed in [11], which uses mathematical analysis to estimate the target position based upon the strengths of the signals heard by neighboring nodes. However, these analytical models were designed for open-air environments where acoustic signal propagation models are known. Unfortunately, acoustic propagation in indoor environments is extremely difficult to model; thus we are not currently able to take advantage of these precise analytical methods. As [11] notes, the complexity of the more sophisticated algorithms is justified only if their accuracy is higher than the node spacing itself. Since this is not the case in our application, we make the same conclusion as [11] that a reasonable approximation to target localization is the location of the node hearing the highest volume.

Thus, our approach (see also [13]) involves each robot filtering its acoustic data and then communicating its volume heard to its local neighbors, as illustrated in Figure 15. Because of noisy data generated by our simple microphones, merely reporting the location of the highest raw volume heard is not sufficient to define the target's current position. We therefore define a *qualifying ratio*, which gives the fraction of time a sensor node must hear the highest volume (relative to its immediate neighbors) during the most recent predefined (short) time interval. If the robot's volume heard over the time interval exceeds the qualifying ratio, it reports its position to the operator control unit (i.e., base station) as the current target position estimate. If there is more than a single sensor robot that exceeds the qualifying ratio (as is sometimes the case when the ratio is less than 0.5), then all such robots will report their positions. If multiple targets are in the environment, all the target locations in the sensor network are reported. If there is a tie in the detected maximum volumes of two neighbors, then both report their positions. If the target is moving, these ties will be broken very quickly. The pseudocode for this distributed acoustic detection algorithm is shown in Table 2.

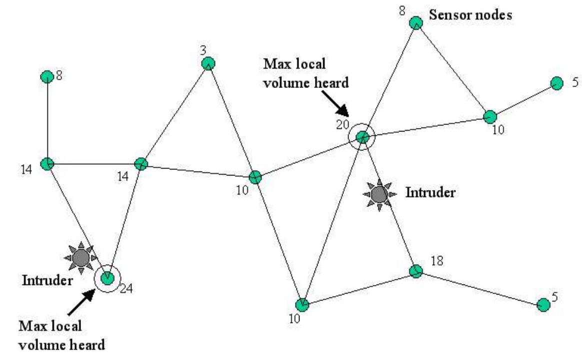


Figure 15: Illustration of how the distributed acoustic sensor net operates. The small circles represent sensor node positions. The edges represent nearby neighbor communications paths. The numbers represent the volume heard at that node. The circled sensor nodes are the local maxima, determined via communications with neighboring nodes.

3.4 Deployment and Detection Results

The deployment and detection approach was validated through extensive experiments performed in the environment shown in Figure 7. The experiments consisted of repeated deployments of 1-3 simple robots per team, as illustrated in multimedia ex-

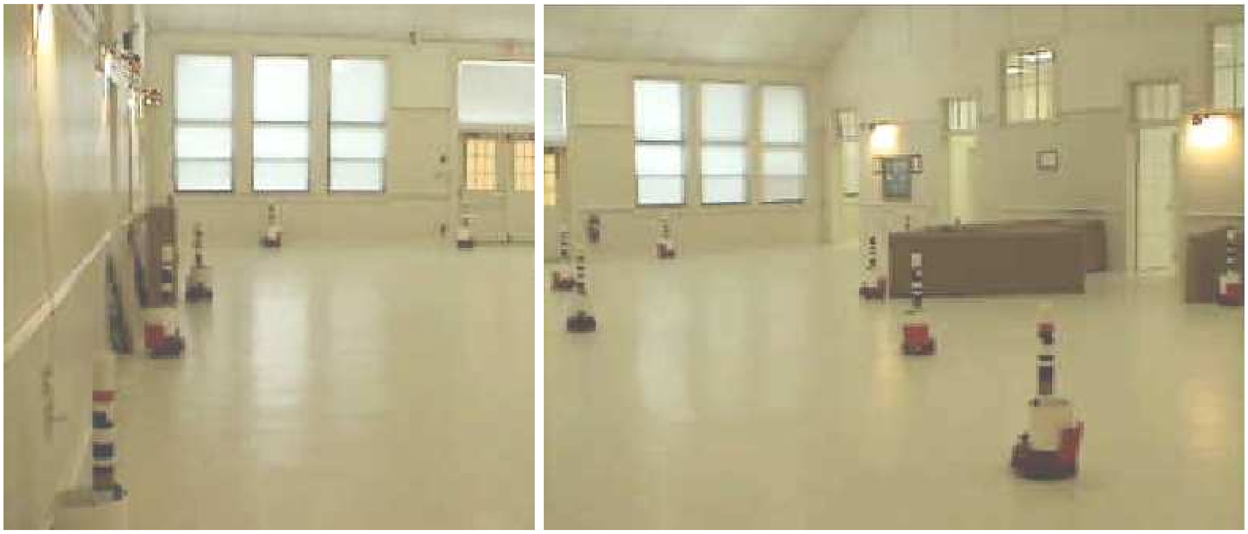


Figure 14: Physical robots deployed according to the results of the autonomous planning process for the environment shown in Figure 9.

tension 2. The experiments were tightly controlled by a set of human evaluators who were not the system developers. Additionally, the experiments were run by human controllers that were allowed access only to laser feedback from the leader robots and a stream of text messages from the robot team members to determine the state of the system. If a deployment failed on one experiment (for example, if a simple robot got caught on an obstacle when trying to follow the leader through a sharp turn), the consequences of that failure were not corrected unless the human controllers could determine from the leader robot's laser feedback that some robot had blocked a passageway. Thus, the data reported here incorporates propagation of error from one experiment to the next. In these experiments, a total of 61 simple robot deployments were attempted.

The color blob tracking algorithm for the chaining on the simple robots proved to be quite robust when operating in uncluttered environments. This algorithm can successfully follow a leading robot as it moves in the environment, up to a 90-degree turn angle. We successfully demonstrated 5 simple robots robustly following a leading leader robot in a chaining behavior. The main limitation to scalability is the tendency of the following behavior to "shave off" corners. Thus, as this tendency propagates through many robots, eventually some simple robot will become lost or caught on obstacles by cutting corners too closely. In our experiments in cluttered spaces, it was difficult for the simple robots to follow the leader when it took many sharp turns in complex environments. The simple color tracking camera also requires a fair amount of light to work properly. We were able to operate very well in open spaces with typical office lighting. Because of the requirement to make several sharp turns in our test environment, the chaining behavior was only found to be robust for teams of 1-2 simple robots in cluttered environments.

The color vision-based fiducial detection software for leader teleoperation was tested independently from the rest of the behaviors to determine its robustness and accuracy as a function of distance and relative position in the leader's field of view. For these independent evaluations, we selected 10 different positions with various lighting and background conditions. Distances between the leader's camera and the simple robot marker varied from 0.5m to 2.5m. When the leader can detect the marker, the determination of the relative pose of the simple robot is very high, with an average error in estimated distance of about 6cm. The primary difficulty is in the leader robot not being able to find the marker, due to distance, unfavorable lighting conditions, or a cluttered visual

Table 2: The Distributed Acoustic Sensing Algorithm.

Distributed Acoustic Sensing Algorithm

While (forever)

- Filter sound – For each sound instant:
  - Subtract out all noise below a predefined volume;
  - Use only sound data within specified frequency range;
  - Average sounds over a short period (approx. 1/3 second)
- Communicate my filtered sound (h) to my nearest neighbors;
- Receive $V[1..n]$ volumes from my n nearest neighbors;
- Update fraction of highest volume heard for most recent time interval. (i.e., if $h > V[i]$ for all i, then fraction increases; else, it decreases.)
- If my fraction of highest volume heard > qualifying ratio,
  - Broadcast my position to the base station as the nearest to the detected target.

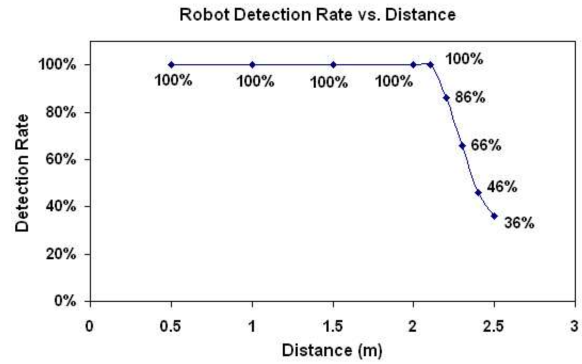


Figure 16: Marker detection accuracy as a function of inter-robot distance.

background. The results of these tests of robustness are shown in Figure 16. The performance is quite good until a distance of about 2.1 meters is reached, due to the limits of the size of the marker and the image resolution. The ability of the leader to detect the marker falls off quickly beyond this distance.

The autonomous teleoperation approach provides accuracy of final simple robot positioning of approximately 30 centimeters, compared to the original planned way-point positions. Since the typical distance between deployed simple robot positions is 2 meters or more, this level of accuracy is suitable for our purposes. Over a set of 36 successful trials, the average time for an individual robot deployment was 132 seconds, with a standard deviation of 45 seconds. The fastest deployment achieved was 65 seconds, while the slowest deployment time was 250 seconds. The variation is typically due to the leader occasionally losing the simple robot's visual marker and having to slowly pan its camera to find it again.

Another metric of evaluation is the percentage of sensor robots successfully deployed (i.e., the ratio of successful deployments to attempted deployments). Our experimental data shows an overall deployment success rate of 60% - 90%, depending upon the environmental characteristics. In other words, for each attempt at deploying a simple robot, 60% - 90% of those robots successfully reached their planned deployment position. The reason for the low end of this success rate is the complexity of our heterogeneous robot system. Our system for simple robot deployment is composed of several non-trivial modules, including localization, path planning, navigation, leader following, visual marker detection, and inter-robot communication. The successful completion of the entire deployment process depends upon the successful completion of all of the system modules while the robots are operating in cluttered environments along complex paths. Additionally, the independent experimentation reported here was especially challenging because we forced the robot team to deal with the consequences of prior deployment failures. Thus, subsequent robot team deployments had to deal with situations such as partially blocked doorways if a prior deployment resulted in a simple robot being caught on the doorway. If all the test runs had been independent, the overall system success rate would certainly have been higher.

Clearly, there are many potential failure modes in such a complex heterogeneous system involving such a large number of robots. The most common failure modes of the system were caused by variable lighting conditions (which could cause the sensor node robots to lose the color blobs, or the leader robots to lose the visual marker for autonomous tele-operation), cluttered environments (which could cause the follower sensor node robots to lose the leader robot amidst many navigational twists and turns), and communications failures (due to delays in multi-hops in the wireless ad-hoc network). To account for these potential subsystem failures, we built extensive fault tolerance into the behavior of the leader robot. Table 3 shows the set of base failure states identified for this system and the implemented recovery action. Using these methods of behavior fault tolerance, the success rate of the leader robots making it back home autonomously in these rigorous experiments was 91% (over 45 trials).

In three separate trials supervised by the independent team, our distributed acoustic sensor network achieved 100% detection of targets in the environment (all targets were localized to the correct room or corridor) with no false positives; these trials involved 3 leaders and up to 35 sensor robots. Clearly, these results show that the planned sensor positions and the distributed acoustic sensor processing worked well for the objectives of these experiments.

4 Discussion and Conclusion

To achieve practical autonomy in large mobile robot teams, one must necessarily adopt a fault-tolerant and cost effective approach to system design. Our ultimate approach incorporated a number of key elements, some designed in from the outset, others learned along the way.

- **Heterogeneity:** by using two distinct classes of robots – one capable but expensive, the other simple but inexpensive – overall system cost has been reduced by nearly an order of magnitude. The down-side of this approach is that algorithm choice is dictated by platform availability, rather than vice-versa. Given the opportunity to repeat this project, we would probably chose a different point in the cost/capability/number trade-space, with a smaller number of slightly more capable robots. This would increase system reliability while saving money on hardware support and software development.

- **Middleware and simulation:** this system integrates a wide range of existing technologies (particle-filter-based localization, configuration space path-planning and obstacle avoidance) with a number of new approaches (manifold maps, mutual observation and follow-the-leader deployment). Recognizing this, we adopted the Player robot server [20] as a form of middleware. Player gives us access to existing implementations of common algorithms, while also providing a set of standard interfaces around which the overall software is structured. Player also gives us access to the Stage and Gazebo simulators [10], which were crucial in the early stages of algorithm development and validation.

- **Communications**: one of the lessons learned during the course of this project was the limitations inherent in off-the-shelf 802.11b wireless technology. When operating with large numbers of devices (greater than three), or over significant indoor distances (greater than 10m), total packet loss can easily exceed 30%. Under these conditions, the effective throughput for TCP-based protocols drops to zero. UDP-based protocols are much more robust, provided the supporting algorithms can tolerate significant data loss.

Table 3: Identified failure states detected by the leader robot and implemented recovery actions.

Failure Type	Fault Recovery Action
Can't reach way-point	Re-plan path.
Lost simple robot	Leave lost robot in wait state and move on to next robot in chain.
Leader robot camera failure	Leave simple robot(s) in wait state, send camera failure feedback to human operator and return home.
Simple robot motor failure	Check if simple robot is close enough to goal; if so, change simple robot state to sensor detection and proceed as if successfully deployed; else, leave simple robot in wait state and proceed to the next simple robot.
Localization drift	Check if simple robot is close enough to goal; if so, change simple robot state to sensor detection and proceed as if successfully deployed; else, leave simple robot in wait state and proceed to the next simple robot.
Can't detect marker	Check if simple robot is close enough to goal; if so, change simple robot state to sensor detection and proceed as if successfully deployed; else, leave simple robot in wait state and proceed to the next simple robot.
Communication failure	Return home.

- **Algorithmic robustness:** given the inevitability of failures during such a large experiment, key algorithms had designed-in robustness. Examples in our project include the use of mutual observation for loop closure (zero errors over seven supervised trials), and leader recovery actions in response to deployment faults (91% successful return to base over 45 trials).

In closing, we note that the multi-robot systems developed for the DARPA SDR project have established new benchmarks for task complexity, team size and experimental rigor. We hope that many of the algorithms and much of the methodology developed for this project will find application in the broader robotics community.

Resources

Raw data from these experiments is available on the Radish web-site [9].

http://radish.sourceforge.net/

Much of the code can be downloaded as part of the Player package:

http://playerstage.sourceforge.net/

Acknowledgments

This work is sponsored in part by DARPA grant 4400057784/SDR (Software for Distributed Robotics) in conjunction with SAIC, Telcordia, the University of Southern California and the University of Tennessee Knoxville.

Thanks to Michael Bailey, Ben Birch, Balajee Kannan, Chris Reardon, Fang Tang, Yifan Tang, Chris Jones, Nate Keonig and Ian Kelly for their significant contributions to this research.

Particular thanks go to John Spofford for his patience and understanding, and to Eric Krotkov and Doug Hackett for their extraordinary efforts in the preparation and supervision of the Fort AP Hill experiments.

A Index to Multimedia Extensions

The multimedia extensions to this article are at:

http://www.ijrr.org.

Extension	Type	Description
1	Video	Multi-robot exploration and mapping
2	Video	Guided robot deployment

References

[1] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, pages 926–939, December 1998.

[2] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proc. of IEEE International Conferenceon Robotics and Automation (ICRA)*, volume 1, pages 476–81, 2000.

[3] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Coding theory framework for target location in distributed sensor networks. In *Proceedings of IEEE International Conference on Information Technology: Coding and Computing*, pages 130–134, 2001.

[4] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*. Morgan Kaufmann Publishers, Inc, July 1990.

[6] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedingsof the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2000.

[7] A. Howard. Multi-robot mapping using manifold representations. In *IEEE International Conference on Robotics and Automation*, pages 4198–4203, New Orleans, Louisiana, Apr 2004.

[8] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5: Proceedings of the Sixth International Symposium on Distributed Autonomous Robotic Systems (DARS 2002)*, pages 299–308. Springer-Verlag, 2002.

[9] A. Howard and N. Roy. Radish: the robotics data set repository. http://radish.sourceforge.net, May 2003.

[10] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004.

[11] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, pages 17–29, March 2002.

[12] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[13] L. E. Parker, B. Birch, and C. Reardon. Indoor target intercept using an acoustic sensor network and dual wavefront path planning. In *Proceedings of IEEE International Symposium on Intelligent Robots and Systems (IROS '03)*, pages 278–283, October 2003.

[14] L. E. Parker, B. Kannan, F. Tang, and M. Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of IEEE International Symposium on Intelligent Robots and Systems (IROS '04)*, pages 1016–1022, 2004.

[15] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee. Pheromone robotics. *Autonomous Robots*, 11(3):319–324, 2001.

[16] N. Roy and G. Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[17] Y. Tang, B. Birch, and L. E. Parker. Planning mobile sensor net deployment for navigationally-challenged sensor nodes. In *IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 172–179, 2004.

[18] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence Journal*, 128(1–2):99–141, 2001.

[20] R. T. Vaughan, B. P. Gerkey, and A. Howard. On device abstractions for portable, reusable robot code. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2121–2427, Las Vegas, Nevada, U.S.A, Oct 2003. (Also Technical Report CRES-03-009).

[21] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Robotic exploration, Brownian motion and electrical resistance. *Lecture Notes in Computer Science*, 1518:116–??, 1998.

[22] B. Yamauchi. Frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*, pages 146–151, 1997.