Centibots: Very large scale distributed robotic teams

Kurt Konolige, Dieter Fox[1], Charlie Ortiz, Andrew Agno, Michael Eriksen, Benson Limketkai[1], Jonathan Ko[1], Benoit Morisset, Dirk Schulz[1], Benjamin Stewart[1], Regis Vincent

SRI International, Artificial Intelligence Center
[1]University of Washington, Department of Computer Science & Engineering

1 100 Robots

We describe the development of Centibots, a framework for very large teams of robots that are able to perceive, explore, plan and collaborate in unknown environments. The Centibots team currently consist of approximately 100 robots (Figure 1). The Centibots team can be deployed in unexplored areas, and can efficiently distribute tasks among themselves; the system also makes use of a mixed initiative mode of interaction in which a user can influence missions as necessary. In contrast to simulation-based systems which abstract away aspects of the environment for the purposes of exploring component technologies, the Centibots design reflects an integrated end-to-end system.

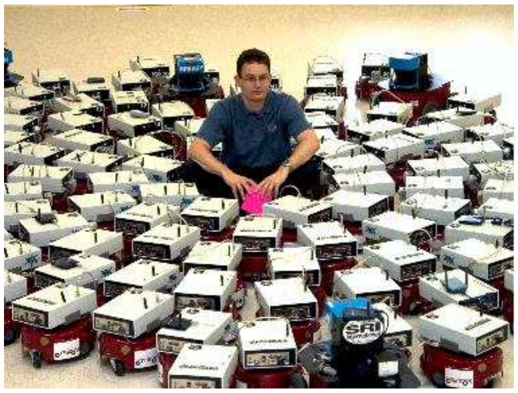
Fig. 1. 100 robots. Four of the robots are Pioneer IIs with SICK laser range-finders. The rest are Amigobots with sonars, a camera and a small PC on top. The OOI is in the hand of one of the authors.

As part of DARPA's Software for Distributed Robotics (SDR) project, the Centibots were tested on a mapping and search mission in a new, unknown environment. This experiment involved deployment of Centibots in three successive stages: (1) a *mapping stage* for the coordinated exploration of the environment while simultaneously constructing a very high accuracy occupancy map using a laser range finder; (2) a *search stage* in which the environment is exhaustively searched for a predefined object of interest (OOI), chosen so that it could be easily distinguished within the environment by its shape and its color; and (3) an *intruder detection stage* in which robots are distributed throughout the environment to "guard" the OOI by continuously searching the environment for human intruders. This stage included recharging a portion of the robots to prove the system could continue indefinitely.

Previous work has largely focused on isolated aspects of our system, including multi-robot exploration [1], architecture [3], task allocation [12], coordination [9], and human interaction [11]. Here we describe the integration of various technologies to achieve an operational robot team, which was tested under rigorous conditions by SDR's outside evaluation team during a final demonstration. The main criteria of the evaluation focussed on the effectiveness of the robot team in performing the mapping and surveillance task (Sections 2.2 and 3.3).[1]

Our approach to multirobot coordination is significantly different between the mapping phase and the subsequent search and surveillance. Mapping is performed with a small number (1-5) of robots working completely autonomously, often out of contact with the base station. Their interactions are tightly focussed on solving a single task, exhaustively mapping an area in the shortest time. We developed specialized algorithms based on utility theory to coordinate the mapping robots, under the condition of an unknown environment, intermittent communication and no centralized planner. In search and surveillance, a much larger number of robots ($\approx$100) must be coordinated, and the tasking is more flexible, e.g., robots can be commanded to watch over a given area. Here, issues of spatial reasoning, task distribution, resource allocation, and user interaction become much more important.

In the following sections, we describe the coordination strategy for mapping and exploration, and give the results of the evaluation for this phase. We then give an account of search and surveillance, along with their results.

2 Distributed Mapping and Exploration

In the mapping phase, multiple robots explore the environment in order to build a map that can be used in the subsequent search and surveillance phases. We developed a decentralized system that goes beyond the state of the art in multi-robot mapping in that it does not depend on reliable communication between robots and makes no assumptions about the robots' relative start locations.

2.1 Overview of Exploration System

Multi-robot Mapping Our technique for multi-robot mapping is based on a representation of *local probabilistic constraints* among robot poses. These constraints arise from robot motion (odometry) and matching laser range-finder scans. Figure 2 shows a laser map along with the trajectory of a robot (gray) and the constraint links (black). The trajectory also represents robot motion links. The optimal position of the poses is the one that maximizes the posterior probability of all the constraints. Although the constraints are nonlinear, there are efficient approximations that work well in practice [4, 7, 8]. Note the fine detail of the laser scan map resulting from this optimization, showing even the thickness of the walls.

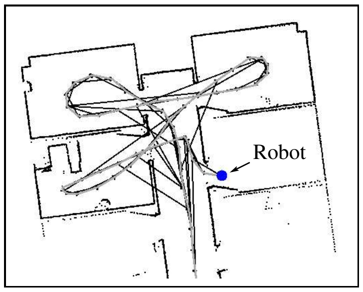

Fig. 2. Robot path (gray) and links among poses. 1m grid shown for size.

[1] Two other teams, one led by SAIC and one by MIT, also underwent the same evaluation process, but as of this writing we do not have access to their results for comparison.

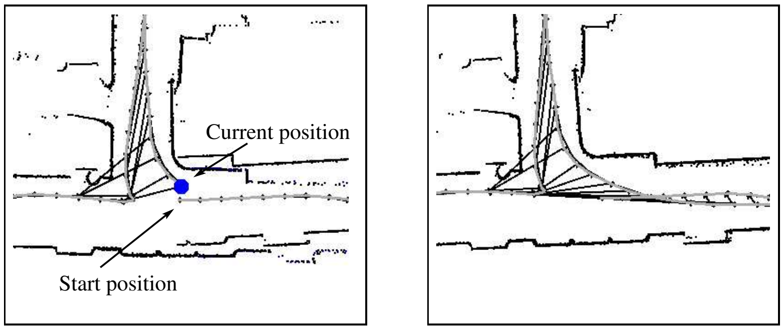

Fig. 3. Pose constraints before (left) and after (right) linking the start and end of a loop.

The constraint network is ideal for integrating map information with uncertain alignment. Consider first the case of closing a loop: a robot returns to a position it has previously visited, but accumulated error causes it to be misaligned (Figure 3, left). Here the robot has traversed an interrupted loop, going out of the top of the figure before coming back. In the right side figure, scan matching has established links with poses at the beginning of the loop, and by optimization the loop can be closed correctly.

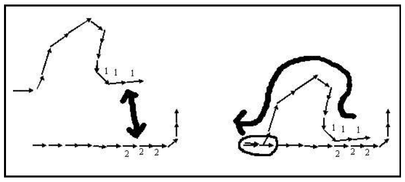

Fig. 4. Zippering two unregistered partial maps. Poses "1" are found to be related to poses "2". After joining, the new poses are added to the first map, starting from the join point; note the additional loop closure (circled region).

The constraint representation naturally facilitates the merging of partial maps built by different robots. For example, the left side of Figure 4 shows the robot poses of two partial maps built independently by two robots, without any notion of where they are with respect to each other. Suppose we can link poses in one map (labeled "1") to poses in the other map ("2"), by some good decision process. Then, we can move the two maps together to register them in the same metric space. Finally, we go through one of the partial maps and add all of its scans to the other map, just as if all scan were collected by a single robot. In this process, the two maps are *zippered* together, adding connections resulting in a globally consistent map.

Abstractly, the zippering process lets us take any partial maps produced by any robots and put them together, once a common location (*colocation*) between their trajectories has been identified (note that colocation is transitive). In order to determine these common locations, we developed an efficient algorithm that sequentially estimates the relative locations between robot pairs as they explore an environment [6]. The approach considers only pairs of robots since the complexity of estimating map matches is exponential in the number of robots considered jointly. For each robot pair, the technique uses an adapted particle filter to estimate the position of one robot in the other robot's partial map. By estimating the posterior over robot positions both inside and outside the partial map, the approach is also able to estimate whether or not there is an overlap between the robots' maps. To accurately determine the overlap probability, we developed a hierarchical Bayesian technique that learns a prior over the *structure* of indoor environments and uses the structural model to estimate the certainty of map matches [2, 10].

Active Multi-robot Colocation and Exploration Virtually any map matching technique can generate false-positive matches, especially in large, highly symmetric environments. A wrong map match between two robots can generate subsequent wrong map matches with other robots. Thus, undoing a wrong match requires considering all other map matches as well. To avoid the complexity resulting from wrong matches, we developed a technique that coordinates robots to *actively* verify whether or not a map match hypothesis is correct. The approach is integrated into a decision-theoretic multi-robot exploration strategy (see [6] for details).

Figure 5(b) shows an example run using our coordination technique. The two robots, `A` and `B`, start from different, unknown locations. Initially, the robots explore on their own. As they explore, each robot estimates the other robot's location in its own map, using the modified particle filter mentioned above. When deciding where to move next, both `A` and `B` consider whether it is better to move to an unexplored area (frontier), or to verify a hypothesis for the other robot's location. At one point, `B` decides to verify a hypothesis for `A`'s location. It sends `A` the message to stop and moves to `A`'s hypothesized location. Upon reaching this location, both robots check the presence of the other robot using their laser range-finders (robots are tagged with highly reflective tape). When they detect each other, their maps are merged using the zippering process described above. From then on, they explore the environment in a coordinated way. If a hypothesis verification fails, on the other hand, then the hypothesis is simply deleted, and all robots keep on exploring.

Our coordination technique works for more than two robots. Multiple robots can share a common map and coordinate to explore and verify hypotheses for the locations of other robots. Since each map merge operation increases the number of robots sharing a common map, team coordination improves over time. Howard et al. [5] also use robot detections to merge maps (and close loops). In contrast to our active colocation technique, their approach is purely passive in that robots have to detect each other coincidentally. Passive map merging can result in significant delays, for example, when one robot follows the path of the other robot and never actually detects it.

Exploration with Limited Communication Robots form so-called exploration clusters, which are groups of robots that share a common map. A team leader robot uses this map to coordinate the other robots. New robots can be added to a cluster, once their relative location with respect to the cluster map is determined. Maps are represented compactly as sets of laser range-scans annotated with robot poses and probabilistic links (scans are recorded only every 50cm). Each robot integrates its observations into its own map, and broadcasts the information to the other robots. While most of the other robots only store this data, the team leader integrates all the sensor information it receives. Thus the team leader has a complete and consistent map representing the data collected by all robots in the cluster. Frequently, this map is broadcast to the other robots, in order to guarantee consistency. The data can be

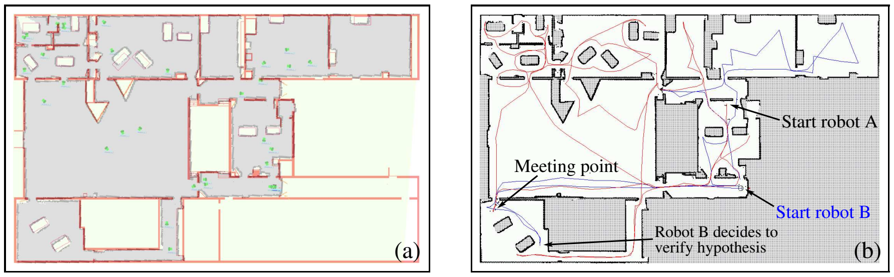

Fig. 5. (a) Map overlayed with the ground truth CAD model of the building. The CAD model was generated by manually measuring the locations and extensions of rooms and objects. (b) Map and paths of an exploration run. The two robots start exploring from different, unknown locations. After finding a good hypothesis for their relative locations, they meet at the meeting point, merge their maps, and continue coordinated exploration.

sent very compactly, since only updated robot poses and links have to be transmitted (scans are already stored by the other robots). The most complex broadcast follows whenever a robot closes a loop, since the optimization operation modifies all robot poses in a map.

Our exploration system achieves robustness to communication loss by enabling every robot to explore the environment on its own. Whenever a robot in an exploration cluster reaches an assigned goal point, it keeps on exploring based on its own map until it receives a new goal point. Thus, if a robot moves outside the communication range of its cluster, it automatically keeps on building its own map until it gets back into communication range. After getting back into communication, robots exchange all the relevant data that was lost. Such a 'sync' operation only involves the communication of rather small data sets. The approach is also robust to loss of the team leader, since any other robot in the cluster can explore on its own or take over the team leader role. In the extreme, if none of the robots can communicate with each other, each robot will explore the environment independently of the other robots. The result will still be a complete map; only built less efficiently.

2.2 Experimental Evaluation of Mapping

The SDR project is unique in having an experimental validation conducted by an outside group. For a week in January 2004, the Centibots were tested at a $650m^2$ building in Ft. A.P. Hill, Virginia. We were tested under controlled conditions, with a single operator in charge of the robot teams via the mixed initiative interface described in the next section. The evaluation criteria for mapping included time to create a map, topological accuracy, and percent of area mapped. Ground truth for mapping was given by a manually constructed map (Figure 5(a)); in fact, the robot's maps were more accurate. Extensive software tuning was circumvented by limiting access to only half of the experimental area during test runs.

The results for four official mapping runs are summarized in Table 1. In all runs, the robots were able to autonoumously generate a highly accurate map of the whole environment. The average mapping time for single robot exploration was 24 minutes; this time was reduced to 18 minutes when using two robots. We also performed

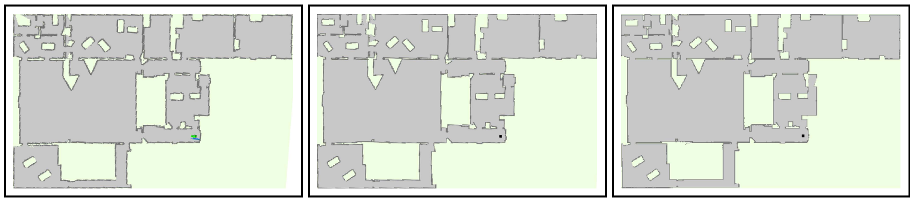
Fig. 6. Maps built during three autonomous exploration runs. The maps look almost identical, even though they were built under very different circumstances; (left to right) by one robot, by two robots starting from the same location, and by two robots starting from different, unknown locations. The similarity between the maps illustrates the robustness of the system and supports our belief that these maps are more accurate than the hand-built map.

additional experimental runs. In one setup, three robots were able to map the area in 15 minutes. Two robots starting from different, unknown locations generated a complete map within 26 minutes (this run is shown in Figure 5(b)). All generated maps looked virtually identically, as shown in Figure 6.

3 Search and Surveillance

In this phase of the mission, the challenge of coordinating the robots becomes more difficult because of the following factors.

- The number of robots is large ($\approx$100).
- The mission goals are not predetermined, and can change during the mission.
- The goals should be specified at a high level, e.g., "search the building."
- The robots must establish and maintain a communications network.
- All robots must be controlled and coordinated by a single operator.

In Centibots there were only two high-level missions: the search for OOI and the protection of the OOI. In both cases the first step was to determine *where* to send the robots to achieve the mission: how the robots can cover the most free space while maintaining other objectives such as a communication backbone. In the Centibots system, a spatial reasoner determines an optimal or near-optimal assignment of robots to positions in the space, while a separate coordination module (the *dispatcher*) implements an assignment that follows the plan, and also monitors and adapts the plan as robots fail.

Many interesting low-level behaviors are built into the search robots, allowing the coordination module to abstract away some of the difficulties of the problem. For example, robots have the ability to stay localized within a map, to navigate to particular positions within the map, and even to perform simple traffic-control behavior such as staying to the right in a corridor. In searching for the OOI, there are behaviors to scan in a circle, to detect the object, and to transmit information about its location. Finally, each robot has a single camera, and algorithms for detecting people as moving objects and reporting their position.

3.1 SPARE

The SPARE component (for SPAtial-REasoning) is a general system dedicated to the spatial allocations in Centibots. SPARE is structured in two parts: spatial representation generation and spatial reasoning.

Occupancy Map $\rightarrow$ Skeleton Map $\rightarrow$ Spatial Reasoning

The output of the mapping phase of the mission is an occupancy map, specifying occupied, unoccupied, and unknown regions. From this, the representation generator constructs an abstract skeleton map of nodes and connections between the nodes. The skeleton map is used by spatial reasoning processes to search for robot locations that maximize their utility for a given task.

Spatial Representation The spatial representation creates an abstraction of the map, called the *topological graph* (*TG*), that has the following characteristics:

- It is topologically correct.
- It is compact.
- All points are reachable by the robots.
- It can be used for a quick but correct path planning computation.
- It contains enough points to find good solutions, but not too many to make the search computationally unfeasable.

To create the *TG*, we first compute a Voronoi diagram (*VD*) from the input occupancy map (a 2D grid). *TG* is then generated from *VD* by filtering, feature identification, and expansion. Figure 7 shows some steps in the generation of *TG*. In general, *TG* consists of a set of nodes and links that cover the area in a manner that facilitates searching and visibility.

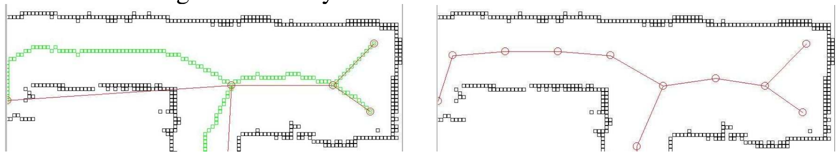

Fig. 7. *VD* components associated with vertices and edges of *TG*, and additional vertices added by the expansion step.

Spatial Reasoning For any given mission, we abstract the problem of placing robots in the environment to the task of assigning robots to nodes of *TG*. We assume all the robots are homogenous, so any robot can fulfill any task requirement. An *assignment* α is a mapping from the nodes of *TG* to 1 or 0 (robot or no robot). It would be a simple modification to change the boolean value to a set of values to account for *classes* of robots.

The *cost* $C(\alpha)$ is a scalar function of the assignment. Since the number of assignments is exponential in the number of nodes, we decompose the cost into subcosts that can be easily calculated, and use an approximate method to determine a good assignment (Section 3.1). In general, we want costs to be local to a single node, or at least to a small neighborhood of nodes, so that incremental optimization algorithms will work well. To this end, we determine the global cost by a summation of smaller cost functions:

$$C(\alpha) = \sum_{i=0}^{p} w_i \sum_{j=0}^{n} c_i(v_j; \alpha) \tag{1}$$

where n is the number of nodes in *TG*, and p is the number of cost functions. The w_i are *weights* that can be changed to reflect the type of mission under consideration. The weights were chosen empirically, to reflect the different priorities in the two missions of searching for the OOI and protecting the OOI.

Note that, potentially, each "local" cost function c_i could involve the whole assignment. In practice, we have developed 10 local cost functions, of which 8 relate to a single node, 1 to a local neighborhood, and one which tries to minimize the number of robots, and so uses the whole assignment.

Finding Good Assignments The problem is to identify the n-tuple $(v_1, v_2, ...v_n)$ that optimizes C with a weight distribution w_i specific to the mission. The search space size is 2^n where n is the number of vertices in *TG* (typically, a few hundred). In such a huge search space, and in the context of our application, our goal was not to find an optimal solution but a satisficing one in a reasonable time (within a few minutes). The quality of a solution is determined by a human expert: a solution is considered good if no misallocation is detected by a human analysing the result. We have found that a standard simulated annealing algorithm works well in finding reasonable solutions.

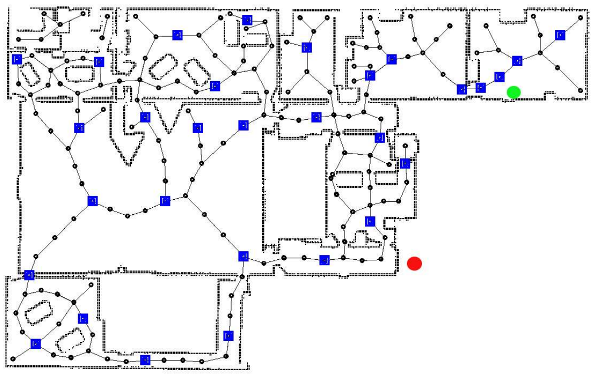

Fig. 8. *TG* shown with superimposed assignment (squares) for a guarding operation. The OOI is in the upper right room; the operator is at the lower right. All rooms are covered for intruder detection, and the comms backbone has a max distance of 10m per robot.

The SPARE system has been tried on over a dozen different maps for several types of tasks. For each map, the computation time does not exceed a few hundred milliseconds. For a given task, the difficulty lies with manual tuning to obtain suitable values for the weight distribution w_i. This tuning is done by trial and error. In practice, once a weight distribution is found for a given task, the weight distribution stays efficient on other maps. For instance, for the searching and protecting tasks performed at the demonstration site in a completely unknown environment, no new tuning of the weight distributions were necessary. Even if the optimality of the solution returned by SPARE is impossible to evaluate formally, the cost diminution between the initial random solution and the final one is considerable (by a factor of 100 on average). Figure 8 shows the *TG* and a guarding assignment for one of the evaluation runs.

3.2 Hierarchical Dispatching

Once we have all the goals generated by SPARE, the operator assigns a mission to a number of robots. The operator can refine the assignment by creating sub-teams to achieve part of mission independently, or let the system coordinate the entire pool of robots. For example, one team could be assigned to create a communications backbone for the search task. Within a team, a manager or *dispatcher* is chosen to assign tasks to individual robots. Robots register with one or more dispatchers and receive an assignment from the set of goals available to the team. A robot can register to several dispatchers, with one preferred. If there are no more goals to be done for the preferred dispatcher, the robot will ask other dispatchers for work, which is a way of load-balancing. When the goal is finished, the robot calls back in to report the completion, says that it is available, and requests a new goal from the dispatcher. The dispatcher can also allocate a set of goals at once for a robot to do, and the robot will only re-contact the dispatcher when it has finished all of them. The dispatcher is designed as a network service that resides physically anywhere on the network. It can run on any team member and only requires local communication within the team. In every experiment with Centibots, the operator only formed one team, therefore our hierarchical dispatching was equivalent to a centralized approach. This approach was quite effective because the team was limited in size (less than 50).

3.3 Experimental Evaluation of Search and Surveillance

For searching, the evaluation criteria were time to locate OOI(s), positional accuracy, and false detections. For the protection stage, the criteria were detection of intruders, and time to first detection. There were four evaluation runs, and the results are shown in the Table 1. They show that the team was highly effective in finding the object and setting up a guard perimeter. Note that we used very simple visual detection hardware and algorithms, since we had limited computational resources on the robots – false and missed detections were a failure of these algorithms, rather than the spatial reasoning and dispatching processes.

Run	Mapping Time	Map Area	Search Robots	Search Time False Pos	Position Error / Topo Error	Intruder Detect	Time to Detect
1	22 min	96%	66	34 min / 0	11 cm / none	75%	8 sec
2	26 min	97%	55	76 min / 1	24 cm / none	50%	8 sec
3	17 min (2 robots)	95%	43	16 min / 0	20 cm / none	25%[2]	8 sec
4	19 min (2 robots)	96%	42	Missed / 2	NA	100%	48 sec
Avg.	21 min	96%	51	30 min / 0.75	14 cm / none	62%	18 sec

Table 1. Results of the 4 evaluation runs.

[2] Caused by a misconfigured tracking filter, fixed before the next run.

4 Conclusions

It is a measure of the state of maturity of mobile robotics that it is possible to field and evaluate a larg teams of robots in the short 18 months of this project. Our Centibots team uses inexpensive COTS mobile robots, cameras, and processors; the algorithms we have developed for mapping, planning and mixed-initiative deployment will work in real-time, in a distributed fashion, with unreliable communications, in an unknown environment. These are some of the toughest real-world conditions yet imposed on a robotics project. We believe that evaluations like this one are an important step to moving AI robotics into the real world.

Acknowledgements

We thank Doug Gage and DARPA for support under the Software for Distributed Robotics Program (Contract #NBCHC020073). We would also like to thank the evaluation team, Erik Krotkov and Douglas Hackett, for their extraordinary efforts in designing and running the demonstration. Part of this research was funded by the NSF under grant number IIS-0093406.

References

1. W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. 2000.
2. D. Fox, J. Ko, K. Konolige, and B. Stewart. A hierarchical Bayesian approach to mobile robot map structure learning. In *International Symposium of Robotics Research*, 2003.
3. Brian P. Gerkey and Maja J Matarić. Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, Taipei, Taiwan, May 2003.
4. J.S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1999.
5. A. Howard, L.E. Parker, and G.S. Sukhatme. The SDR experience: Experiments with a large-scale heterogenous mobile robot team. In *Proc. of the International Symposium on Experimental Robotics*, 2004.
6. J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.
7. K. Konolige. Large-scale map-making. In *Proc. AAAI*, 2004.
8. F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.
9. P. Modi, W. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of Autonomous Agents and Multi-Agent Systems, 2003.*, 2003.
10. B. Stewart, J. Ko, D. Fox, and K. Konolige. The revisiting problem in mobile robot map building: A hierarchical Bayesian approach. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 2003.
11. A. Tews, M. Mataric, and G. Sukhatme. A scalable approach to human-robot interaction. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2003.
12. Osher Yadgar, Sarit Kraus, and Charles Ortiz. Hierarchical organizations for realtime large-scale task and team environments. In *AAMAS*, 2002.