In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS2003)*
pp. 1939-1944, Las Vegas, Nevada, October 27-31, 2003

Sensor Network-based Multi-Robot Task Allocation

Maxim A. Batalin and Gaurav S. Sukhatme
Robotic Embedded Systems Laboratory
Center for Robotics and Embedded Systems
Computer Science Department
University of Southern California
Los Angeles, CA 90089, USA
maxim@robotics.usc.edu, gaurav@usc.edu

***Abstract*— We present DINTA, Distributed In-network Task Allocation - a novel paradigm for multi-robot task allocation (MRTA) where tasks are allocated implicitly to robots by a pre-deployed, static sensor network. Experimental results with a simulated alarm scenario show that our approach is able to compute solutions to the MRTA problem in a distributed fashion. We compared our approach to a strategy where robots use the deployed sensor network for efficient exploration. The data show that our approach outperforms such an 'exploration-only' algorithm. The data also provide evidence that the proposed algorithm is more stable than the 'exploration-only' algorithm.**

I. INTRODUCTION

This paper presents DINTA, Distributed In-network Task Allocation - a novel paradigm for multi-robot task allocation (MRTA) where tasks are allocated implicitly to robots by a pre-deployed, static sensor network. In prior work [1], [2] we have developed an algorithm for the deployment, and maintenance of such a static network by robots. We have also developed algorithms for exploration and navigation [1], [2] where robots use the deployed network to efficiently explore their environment and navigate to a designated goal. In this paper, we assume the network is pre-deployed (through means outlined in [1], [2]), and robots have to perform spatially and temporally distributed tasks efficiently. Our solution is to allow the process of task allocation to occur in the static network through distributed computation and implicit assignment of robots to tasks.

We are generally interested in the mutually beneficial collaboration between mobile robots and the static network. The underlying principle in interaction between the network and robots is: the network serves as the communication, sensing and computation medium for the robots, whereas the robots provide actuation, which is used among other things for network management and updating the network state.

We study a particular experimental scenario, emergency handling, as an experimental substrate. In prior work [3], we have used a similar scenario to study the role of opportunism vs. commitment MRTA. In our experimental scenario, events in the environment trigger alarms. An alarm is spatially focused, but has temporal extent (i.e. it remains on until it is turned off by a robot). Alarms are detected by nodes in the static network. The task of the team of robots is to turn off the alarms by notionally responding to the emergency signaled by each alarm. This is done by a robot navigating to the location of the alarm which causes the alarm to shut off. The goal is to minimize the cumulative alarm *OnTime* across all alarms, over the duration of the entire experiment. Each Alarm's *OnTime* is computed as the difference between the time the alarm was turned off by a robot and the time the alarm was detected by one of the nodes of the network. We make the following assumptions:

1) The sensor network is predeployed into the environment using algorithm given in [2].
2) In addition to deploying the network nodes, the deployment algorithm also computes the distribution of transitional probabilities $P(s'|s,a)$ from network node s to s', when the robot executes action a [1].
3) An alarm requires at least one robot to service it. To turn of an alarm, a robot needs to appear in its vicinity. Thus, the handling of the alarm is purely notional since that is not our focus here.

More telling, perhaps, is a list of what we do not assume:

1) The robots do not have a pre-decided environment map or access to GPS.
2) The environment is not required to be static.
3) The robots do not perform localization or mapping.

The key result of this work is that our approach statistically outperforms an 'exploration-only' algorithm. In addition, the obtained results show that the proposed algorithm is more stable than the 'exploration only' algorithm.

II. RELATED WORK

Our work is closely related to the body of literature on using markers to aid mobile robot navigation. This idea has received attention in coverage and exploration [1], [2], [4], [5], [6], and navigation [1], [7], [8]. Ant-like trail laying algorithms [7], [6] consider a special case of the *marker* deployment approaches - when the distance between the two consecutive *markers* is small. Therefore a *trail* is formed that the robots can follow and cover the environment and/or navigate. In these cases, no inter-marker communication is necessary, indeed the markers are passive 'read-only' devices.

In [4], [5] the problem of graph coverage using a few *markers* is considered. In both cases the authors study the problem of *dynamic* single robot coverage on an environment consisting of nodes and edges (a graph). The key result was that the ability to tag a limited number of nodes (in some cases only one node) with unique *markers* dramatically improved the cover time.

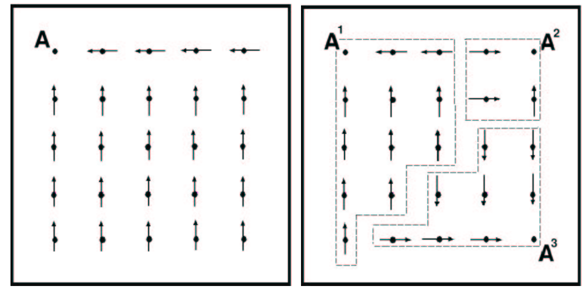
The problem of multi-robot task allocation (MRTA) has received considerable attention. For an overview and comparison of the key MRTA architectures see [9], which subdivides MRTA architectures into behavior-based and auction-based. For example, ALLIANCE [10] is a behavior-based architecture that considers all tasks for (re)assignment at every iteration based on robots' utility. Utility is computed by measures of acquiescence and impatience. Broadcast of Local Eligibility [11] is also a behavior-based approach, with fixed-priority tasks. For every task there exists a behavior capable of executing the task and estimating the utility of robot executing the task. Auction-based approaches include the M+ system [12] and Murdoch [13]. Both systems rely on the Contract Net Protocol (CNP) that puts available tasks for auction, and candidate robots make 'bids' that are their task-specific utility estimates. The highest bidder (i.e., the best-fit robot) wins a contract for the task and proceeds to execute it.

DINTA differs from the above MRTA approaches in the following ways:

1) DINTA relies on a static *network*, thus communication, sensing and computation are distributed.
2) The utilities of task assignments are propagated and computed by the network based on purely local communication between network nodes.
3) The system does not require mobile robots to be within communication range of each other. The network is used for propagating messages between robots.
4) The system does not place a limitation on the number of robots. There is no computation or communication overhead associated with increasing the number of robots.
5) The system does not require one robot to recognize another robot.

III. APPROACH

The basic idea of DINTA is that given a set of alarm-weight pairs (a_i, w_i) detected by the network, every node k in the network computes a suggested direction that a robot should take if in the vicinity of k. This computation results in a direction which maximizes the net utility of the robot. The weight w_i is an abstraction, which is a scalar representation for several parameters like priority, magnitude, time (older alarms should be served first), etc. The ensemble of suggested directions computed over all nodes is called a navigation field. An adaptive distributed value iteration algorithm is used to compute the navigation field. An example of a navigation field for one and three alarms is shown in Figure 1.



(a) One alarm (b) Three alarms; A^1, A^2 and A^3

Fig. 1. Examples of navigation field computed.

It may be noted that an alternative approach for the construction of a navigation field has been proposed in the sensor network literature [8]. Instead of value iteration [8] uses potential fields and the hop count to compute the magnitude of the directional vectors.

A. Philosophy

The general idea of DINTA is to use a static network and mobile robots cooperatively. The network provides a 'sensor' that is 'stretched' over the environment and thus widens the range of applications for groups of robots that do not cover the whole environment - 'can't be everywhere at the same time'. Thus, an alarm can be detected even though no robot is within sensor range. In addition, mobile robots can communicate through the static network even if they are not within communication range of each other. The other benefit of using the network is distributed computation. First, there is no redundant computation (on each separate robot). Second, since every node of the network updates its state based only on the state of its neighbors and robots in the vicinity, the system is scalable. Third, utilities are computed in the network distributively and propagated from the alarm (the goal state). Another benefit is that the robots used can be very simple since they do not need to localize and map the environment - they navigate by listening to the suggestions from the sensor network.

The DINTA approach has two subsystems - *Coverage/Exploration* and *Alarm Response*. If no alarms are detected, the system operates in *Coverage/Exploration* [1], [2] mode. In this mode, the navigation field computed by the network, causes the robots to patrol the environment. If, on the other hand, an alarm is detected, the system switches to the *Alarm Response* mode where the navigation field computed by the network guides the robots to turn off alarms, thereby implicitly solving the MRTA problem.

B. Coverage/Exploration

The *Coverage/Exploration* subsystem is described in [2]. The approach uses interaction between the two entities: the markers (nodes of the network) and the mobile robots. The task of each marker is to recommend a locally preferred direction of exploration for the robot within its communication range. Thus each marker acts as a local signpost telling a robot which direction to go next. However, the robot treats this information as a recommendation, and combines this advice with local range sensing to make a decision about which direction to actually pursue.

Each marker has a state associated with the four cardinal directions (South, East, North, West). The choice of four directions is arbitrary. It implies that the marker is equipped with a 2 bit compass. For each direction, the marker maintains a state and a counter. A state can be either `OPEN` or `EXPLORED`, signifying whether the particular direction was explored by the robot previously. A counter C is associated with each direction; it stores the time since that particular direction was last explored. When the robot is in the vicinity of a marker, the marker emits a suggested direction the robot should take. This implies that the robot's compass and the marker's compass agree locally on their measurement of direction. Given the coarse coding of direction we have chosen, this is not a problem in realistic settings. The algorithm used by the markers to compute the suggested direction is simple. All `OPEN` directions are recommended first (in order from South to West), followed by the `EXPLORED` directions with largest last update value (largest value of C).

The robot remembers the identification of the marker it heard most recently. If, during motion, a new marker is heard, (i.e. the robot moved to the communication zone of a different marker), the robot analyzes the data messages received from the current marker and orients itself along the suggested direction. In addition, the robot sends an update message to the marker telling it to mark the direction from which the robot approached the beacon as `EXPLORED`. This ensures that the direction of recent approach will not be recommended soon. After the robot has been oriented in a new direction, it checks its range sensor for obstacles. If the scan does not return any obstacles, the robot proceeds in the suggested direction, while sending an update beacon message (upon receiving this message the current marker updates the state of corresponding direction to `EXPLORED` and resets the corresponding C

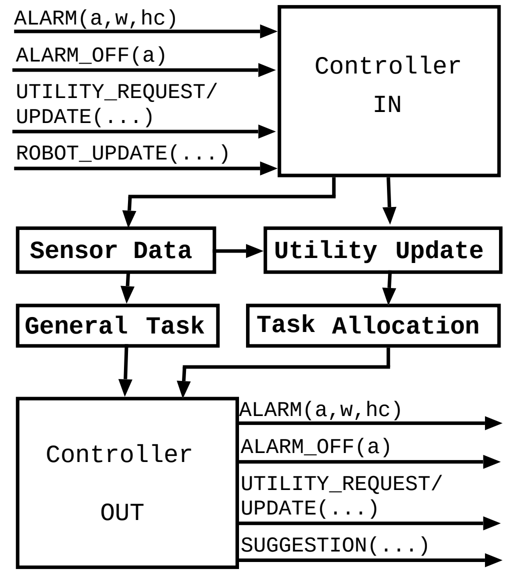


Fig. 2. Generalized Node Architecture.

value). If, however, the suggested direction is obstructed (something is in the way), robot sends a broadcast message updating the marker with this information and requests a new suggested direction. For the details of the approach and theoretical analysis the reader is referred to [2].

[2] shows that the asymptotic performance of *Coverage/Exploration* is between the performance of breadth-first search $O(n)$ and $O(n \ln n)$, where n is the number of nodes in the network. Simulations show correspondence in performance to theoretical results and that this algorithm constantly outperforms a random walk ($O(n^2)$).

C. Alarm Response

Figure 2 shows the data flow on a network node. If a node receives an *ALARM* message with identification a of the node that detected the alarm, weight w (estimation of the alarm's importance) and *hop count* h (estimation of how far away node a is), the alarm is placed on the list L of currently active alarms according to its utility U (`Utility Update` block). We define the utility as the ratio $U = \frac{w}{h}$. This ratio helps node to decide which alarm has the highest priority (i.e utility). In other words, since only one global field is maintained (every node computes one assignment direction), the nodes have to decide which alarm would give larger reward to the robot if it would start at the node. Every node maintains a *current alarm* variable, which is the element of L with largest utility. If the *current alarm* changes, the `Task Allocation` block computes a new task assignment for a robot (discussed next) and reroutes the alarm message with incremented hop count to neighboring nodes. In the global perspective, prioritizing between the tasks accord-

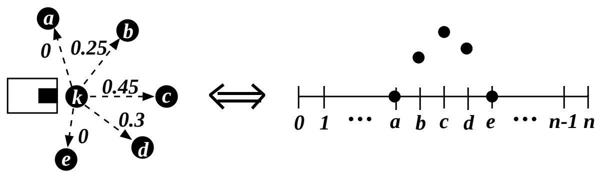


Fig. 3. An example of a discrete probability distribution of node k for direction (action) "East"(i.e. right).

ing to their utility value results in creation of multiple superimposed navigation fields (for example three alarms case of Figure 1). Note that if `Sensor Data` block indicates that an alarm is detected by the node itself, then the node initiates a message *ALARM(thisNodeID, w, 0)*. `General Task` block represents the approach of subsection III-B, which, in case L is empty and based on the current state, sensor data and robot update data, computes the suggested direction of exploration.

The task allocation problem for *emergency handling* can be formulated as guiding robots towards a specific goal state (alarm). Hence, the problem can be considered as the problem of navigation. We assume that the network is deployed and every node has a discrete probability distribution of the transitional probability $P(s'|s_C, a)$ (probability of arriving at node s' given that the robot started at node s_C and commanded an action a). The reader is referred to [1] for detailed discussion on how such distributions can be obtained. Figure 3 shows a typical discrete probability distribution for a node per action (direction). Note that in practice the probability mass is distributed around neighboring nodes and zero otherwise.

Note that the state the robot transitions to depends only on the current state and action. We model the navigation problem as a Markov Decision Process [14]. To compute the best action at a given node the value iteration [15] can be used on the set of nodes $S - s_g$, where s_g is the goal state. The general idea behind value iteration is to compute the values (or utilities) for every state and then pick the actions that yield a path towards the goal with maximum expected value. The value is incrementally computed:

$$V_{t+1}(s) = C(s,a) + \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s,a) \times V_t(s') \quad (1)$$

where $C(s,a)$ is the cost associated with moving to the next state (node). Usually the cost is chosen to be a negative number which is smaller than $\frac{-(minimalreward)}{k}$, where k is the number of nodes. The rationale is that the robot should 'pay' for taking an action (otherwise any path that the robot might take would have the same value), however, the cost should not be too big (otherwise the robot might prefer to stay at the same state).

Initially the value of the goal state is set to the weight parameter and of the other states to 0. Given the values, an *action policy* is computed for every state s as follows:

$$\pi(s) = \arg\max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s,a) \times V(s'); \quad (2)$$

The value iteration algorithm assumes central computation. We augment the traditional algorithm for usage in sensor network. The idea is that every node in the network updates its value and computes the optimal task assignment (navigation action) for a robot in its vicinity on its own. Once the *current alarm* has been changed, every node starts the computation of the optimal task assignment by updating values according to equation 1. Note that the values of neighboring nodes are needed as well, hence, the node queries its neighbors for corresponding values. Note also that Distributed Value Iteration is a Dynamic Programming problem and the general solution to asynchronous Dynamic Programming was proposed in [16].

After the values are computed, every node computes an optimal policy for itself according to equation 2. Neighboring nodes are queried once again for the final value. The computed optimal action is stored at each marker and is sent as a *SUGGESTION* message, to any robots in the vicinity.

Note that the action policy computation is done only once per alarm, and does not need to be recomputed. Also, note that value update equations have to be executed until the desired accuracy is achieved. For practical reasons the accuracy in our algorithm is set to 10^{-3}, which requires a reasonable number of executions of the value update equation per state (approx. 20) and thus, the list of values that every node needs to store is small (20). Since the computation and memory requirements are small it is possible to implement this approach on the real marker device that we are using (the Mote [17]).

IV. SIMULATION EXPERIMENTS

In our experiments we used the Player/Stage [18], [19] simulation engine populated with a simulated Pioneer 2DX mobile robots equipped with $180°$ field-of-view planar laser range finders (used for obstacle avoidance), wireless communication and a mote base station (to communicate with the Motes, used as network nodes). A network of 25 Motes was predeployed in a test environment. The communication range of motes and robots was set to approximately 4 meters. The task of the team of robots is to serve emergencies by navigating towards the point of alarm and minimize the cumulative alarm *OnTime*. Alarm's *OnTime* is computed as difference between the time alarm was served by a robot and the time alarm was detected by one of the nodes of the sensor network.

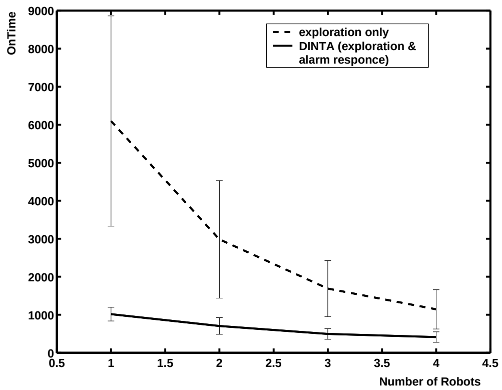


Fig. 4. Comparison between implementation of DINTA and efficient exploration

We conducted experiments in an environment of $576m^2$ with robot group sizes varying from 1 to 4, 10 trials per group. For experiments the schedule of 10 alarms was drawn (time-wise) from a Poisson distribution, with uniformly distributed nodes that detected the alarm and the weights that were assigned to the alarms. The parameter of Poisson distribution was set to $\lambda = \frac{1}{60}$, which means that the expected number of alarms is 10 in 600 seconds.

The implementation of the proposed approach in simulation proceeds as follows. If there are no alarms detected in the environment, then the robots execute exploration algorithm of [1], [2]. If an alarm is detected, the network computes task assignments (navigation field in our case). Once the tasks are computed at every node, the robots change from *EXPLORATION* to *ALARM* mode, and traverse the directions suggested by the network. When one robot reaches an alarm node, the robot injects an ALARM-OFF message into the network, which when received by any node causes removal of the alarm from its list L. Note that if there are multiple alarms active at the same time, a superposition of several navigation fields is produced, in which case robots might follow different paths depending on the portion of the environment in which they are located. Examples of the navigation fields are shown on Figure 1.

Figure 4 shows the *OnTime* comparison for exploration only and the approach proposed in this paper (exploration and alarm response). Clearly, the proposed approach outperforms the exploration only algorithm even though as environment becomes saturated with robots, the difference becomes smaller. The difference is statistically significant (the T-test p-value is less than 10^{-4} for every pair in data set).

Moreover, the performance of DINTA is stable (small and constant variance) whereas variances produced by exploration approach change drastically and reduce as environment becomes saturated with robots.

The proposed first-cut implementation of DINTA does not make explicit assignments of tasks to robot or specific robot subgroups, which may result in suboptimal behavior. Consider the case when the robots are cluttered in one region and therefore, can all be attracted towards the same alarm. Although, in practice this phenomenon occurs rarely, since exploration behavior of robots strives to disperse robots in the environment. We are developing a DINTA implementation which allows the computation of multiple task assignments at every node (a task per robot or robot subgroup). The space and time requirements for this implementation are linear in the number of alarms, which makes it realistic for implementation on our target node platform (the Mote). In addition this approach would allow online grouping of robots for more complex tasks.

V. CONCLUSION AND FUTURE WORK

In this paper we introduced DINTA: Distributed In-Network Task Allocation for solving the MRTA problem. DINTA allows us to combine the benefits of a sensor network with mobility and functionality of robots. The system compute task assignments distributively in-network while, at the same time, providing a virtual sensor and communication device that 'extends' throughout the whole environment and has obvious benefit over traditional MRTA approaches. The fundamental assumption, though, is the existence of the sensor network. However, [1], [2] show that given set of markers large enough, a sensor network can be deployed into an environment and maintained by the robots.

There are several advantages in using DINTA as opposed to other MRTA approaches. The sensor network allows robot to detect a goal (alarm) even though the goal is not in robot's sensor range. In addition, mobile robots can use sensor network to relay messages if they are not in the communication range of each other. One of the other benefits of using DINTA is distributed in-network computation, which 1. avoids redundant computation by updating the state of a node based only on the state of its neighbors and robots in the vicinity (scalability), 2. computes utilities in network distributively and propagate from the goal state (alarm). Another benefit is the ease of determining relative distance to the goal (for determining utilities) by considering 'hop counts' from the goal state. Note also that robots implementing DINTA can be quite simple - they do not need to localize and map the environment - they can navigate by listening to the suggestions from the sensor network.

In future work we plan to extend the current implementation of DINTA to allow explicit assignment of tasks to robots or robot subgroups, which will improve the performance in some cases and would allow more complex task assignments as well as group formations.

We also plan to conduct further experiments both in simulation and hardware in varying environments, with tasks of varying complexity, requiring different numbers of robots. A system would have to assign not only a task, but also combine robots in a group if a task requires participation of several robots.

VI. ACKNOWLEDGMENTS

This work is supported in part by NSF grants ANI-0082498, IIS-0133947, CCR-0120778, and EIA-0121141.

VII. REFERENCES

[1] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," in *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03*, Palo Alto, 2003, pp. 376–391.

[2] M. A. Batalin and G. S. Sukhatme, "Efficient exploration without localization," in *To appear in Proc. of IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, 2003.

[3] E. H. Ostergard, M. J. Mataric, and G. S. Sukhatme, "Distributed multi-robot task allocation for emergency handling.," in *In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2001, pp. 821–826.

[4] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," in *Annual ACM Symposium on Theory of Computing (STOC '98)*, 1998, pp. 269–278.

[5] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," in *IEEE Transactions on Robotics and Automation, 7-6*, 1991.

[6] J. Svennebring and S. Koenig, "Trail-laying robots for robust terrain coverage," in *To appear in Proc. of IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, 2003.

[7] R.T Vaughan, K. Stoy, G. S. Sukhatme, and M.J. Mataric, "Lost: Localization-space trails for robot teams," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 796–812, 2002.

[8] Qun Li, Michael DeRosa, and Daniela Rus, "Distributed algorithms for guiding navigation across a sensor network," Tech. Rep., Dartmouth, Computer Science Department, October 2002.

[9] B. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *To appear in Proceedings of the IEEE International Conference on Robotics an Automation (ICRA03)*, Taipei, Taiwan, 2003.

[10] L. E. Parker, "Alliance: An architecture for fault-tolerant multi-robot cooperation.," in *IEEE Transactions on Robotics and Automation*, 1998, vol. 14, pp. 220–240.

[11] B. B. Werger and M. J. Mataric, *Distributed Autonomous Robotic Systems 4*, chapter Broadcast of Local Eligibility for Multi-Target Observation, pp. 347–356, Springer-Verlag, 2000.

[12] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement.," in *Proc. of IEEE International Conferenceon Robotics and Automation (ICRA)*, 2000, pp. 293–298.

[13] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination.," in *IEEE Transactions on Robotics and Automation*, 2002, vol. 18, pp. 758–768.

[14] D. J. White, *Markov Decision Process*, John Wiley & Sons, West Sussex, England, 1993.

[15] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains," Tech. Rep. CMU-CS-93-106, Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213, December 1992.

[16] D. P. Bertsekas, "Distributed dynamic programming," *IEEE Trans. Automatic Control*, vol. AC-27, no. 3, pp. 610–616, 1982.

[17] K. S. J. Pister, J. M. Kahn, and B. E. Boser, "Smart dust: Wireless networks of millimeter-scale sensor nodes," *Electronics Research Laboratory Research Summary*, 1999.

[18] B. P. Gerkey, R.T. Vaughan, K. Stoy, A. Howard, G.S. Sukhatme, and M.J. Mataric, "Most valuable player: A robot device server for distributed control," in *IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems (IROS)*, Wailea, Hawaii, 2001.

[19] R.T. Vaughan, "Stage: a multiple robot simulator," Tech. Rep. IRIS-00-393, Institute for Robotics and Intelligent Systems, University of Southern California, 2000.