Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination

Brian P. Gerkey · Maja J Matarić
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781, USA
{bgerkey|mataric}@cs.usc.edu

Abstract

We present a distributed planar object manipulation algorithm inspired by human behavior. The system, which we call pusher-watcher, *enables the cooperative manipulation of large objects by teams of autonomous mobile robots. The robots are not equipped with gripping devices, but instead move objects by pushing against them. The* pusher *robots have no global positioning information, and cannot see over the object; thus a* watcher *robot has the responsibility for leading the team (and object) to the goal, which only it can perceive. The system is entirely distributed, with each robot under local control. Through the use of* MURDOCH, *an auction-based resource-centric general purpose task-allocation framework, roles in the team are automatically assigned in an efficient manner. Further, robot failures are easily tolerated and, when possible, automatically recovered. We present results and analysis from a battery of experiments with* pusher-watcher *implemented on a group of Pioneer 2 mobile robots.*

1 Introduction

Multi-robot coordination is a complex control problem, especially in *tightly-coupled* tasks that involve a mutual dependence of the robots on each others' performance. The problem may be made even more complex through the use of heterogeneous robots, with different capabilities. How can groups of such heterogeneous robots coordinate their behavior so as to execute tightly-coupled tasks?

In previous work [3], we proposed a partial answer to this question in the form of MURDOCH, a general-purpose task-allocation system. MURDOCH was designed for use on physically embodied robots living and working in noisy, dynamic environments in which they have little information and even less control. In this paper, we apply MURDOCH to the particularly difficult problem of multi-robot box-pushing. Using MURDOCH, we have implemented a distributed control system, called pusher-watcher, that enables a team of heterogeneous robots to cooperatively relocate a large box to a specified goal, despite having no global position information and no detailed model of the box or its physical properties. We evaluate the system in five sets of experiments and present quantitative results and analysis.

2 Related Work

Box-pushing has long been one of the canonical task domains for mobile robot researchers. At one extreme, [5] describes a swarm-like method for moving a large box with many small, locally controlled robots; the system could fairly be described as *emergent*. At the other extreme are the planner-based master-slave pushing system described in [8] and the centralized formation-based system described in [12]. A similarly deliberative approach is described in [1]; there, the authors focus on the analysis of various two-robot pushing protocols with regard to information requirements. Some middle ground is found by the two-robot behavior-based approach presented in [6], with an emphasis on the robots' learning policies to enable effective cooperation. In [9], a fault-tolerant two-robot box-pushing system is developed, and proof-of-concept demonstrations are given. More recently, a method for single-robot box-pushing through an obstacle field (in the context of robot soccer) is given in [2].

With regard to the pushing control system itself, the work we present in this paper is most similar to the pusher-steerer protocol in [1], the three-robot formations of [12] and, to a lesser extent, the pusher-supervisor system in [8]. However, none of these approaches made any provision for robot failures. Of the other three multi-robot systems, only [6] is goal-directed, and in that case, both pushing robots could directly perceive the goal, somewhat reducing the need for cooperation.

A great deal of work has also been reported on multi-agent coordination systems, though the agents themselves are seldom physically embodied. Notable exceptions are BLE [13], and the ALLIANCE architecture [10], both of which have been applied to a multi-target tracking task with groups of robots. ALLIANCE was also applied to a box-pushing task [9], but the system runs open-loop, whereas we have closed the control loop through the use of our watcher, with the side-effect of adding new op-
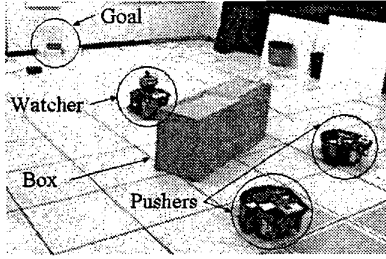
464

Figure 1: *Our experimental box-pushing setup. The task is for the* pushers *to move the box to the goal with the help of the* watcher.

portunities for cooperation. Further, while ALLIANCE relies on an expertly-built structure of interconnected *motivational behaviors*, MURDOCH [3] provides a general-purpose, resource-centric, fitness-based task-allocation system that is a variant of the Contract Net Protocol [11] and is independent of the robots' internal control systems. The use of MURDOCH requires only that each robot implement the external interface through which auctions for tasks are conducted. Thus, for example, one robot in a team may be controlled in a behavior-based manner while another may employ a deliberative planner.

3 Algorithm

The task we address is cooperatively moving a box, large relative to the size of the robots, from some initial location to a designated goal location. In solving this problem, we take inspiration from human coordination behavior commonly observed when people move large pieces of furniture. If the people who are pushing or carrying the piece of furniture cannot see where they are going, another person stands between the carried object and the goal and periodically directs them. This "watcher" can see both the current position of the object and the goal, and thus can compute the error signal, perhaps in the form of a correction angle, that can be communicated to the "pushers"[1].

In the cooperative mobile robot domain, we formally define this problem with a set of constraints:

- The box is large compared to the robots[2], and the robots can only move the box by pushing through frictional contact.

- The pushing robots cannot in general perceive the goal due to occlusion by the box.

- The box can be sensed by the robots that are to push it and the goal can be sensed by the robot that is to act

---

[1]Actually, the watcher likely will not communicate the raw angle, but rather some higher-level command, such as "push more on the right"; we do the same (see Section 4).

[2]Specifically, the intended contact surfaces should allow at least two robots to be pushing simultaneously.

as watcher. Additionally, the watcher robot can sense the position and orientation of the box relative to its own pose[3]. In our implementation, the box and goal are brightly colored and the robots use color cameras to sense them; the watcher obtains the orientation of the box with a laser range-finder.

- There is an obstacle-free path between the box's initial location and the goal that is wide enough for the box and robots to pass (i.e., we do not consider negotiating obstacles in a coordinated fashion, only as part of individual low-level control).

Given these constraints, we have implemented a multi-robot control system, called pusher-watcher, that is similar to the common human solution for this task. As shown in Figure 1, two robots act as pushers, and a third performs the watcher role. The pushers can see the box, and the watcher can see the goal. In addition, the watcher, while servoing on the goal, can accurately perceive (using a scanning laser range-finder) the angular error of the box's orientation with respect to the path from the box to the goal. Our aim, then, is to rotate the box until that angular error is zero (i.e., the box is orthogonal to the path to the goal), while simultaneously translating it toward the goal.

In order to implement this algorithm, we must first determine the pushers' velocities. To this end, as shown in Figure 2, we model the box and watcher together as a rigid body; we attach an imaginary link between the center of rotation of the watcher, C, and the center of the side of the box, orthogonal to the box. The box and link together rotate freely about C. The pushers are assumed to be point velocities that act on this rigid body (for simplicity, we disregard mass and acceleration). At each point in time, the watcher is rotated away from the normal to the box by an angle θ, and is moving toward the goal with a velocity v_w. These translational and angular velocities of the box will then be governed, respectively, by two simple equations:

$$v_t = v_w \cos \theta$$

$$(d + w)\dot{\theta} = v_w \sin \theta$$

After solving for $\dot{\theta}$, we can distribute it differentially to the two pushing points:

$$v_d = \frac{l}{2} \frac{v_w}{(d + w)} \sin \theta$$

Now, we compute the two pusher velocities:

$$v_p = v_t \pm v_d$$

---

[3]This is not to say that either the box or goal need be visible at all times, but rather simply that they are observable phenomena. Each robot will, under local control, automatically reacquire the box or goal in the event that visual contact is lost.
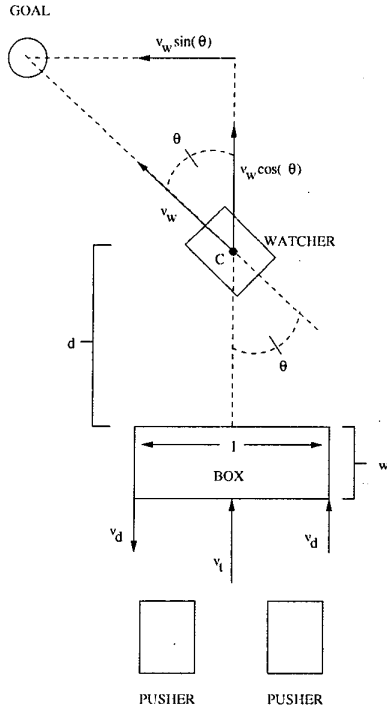
GOAL

Figure 2: *The model used to derive the pushing velocities for moving the box along the desired trajectory.*

The velocities given by v_p, if applied continuously at either end of the box, will ensure that a constant distance is maintained between the box and the `watcher` and that the angular error θ tends toward zero. The actual trajectory will be a curve that approaches the path to the goal with the box's orientation tending toward orthogonality with respect to that path.

The astute reader will note that in certain situations (e.g., when θ is large), this control law can yield negative pushing velocities; such implementation-specific issues are addressed in Section 5.

4 Coordination Through Communication

As previously mentioned, we have implemented the `pusher-watcher` system using the task-allocation facilities provided by MURDOCH. We now give a brief overview of MURDOCH and how it is applied to the multi-robot box-pushing problem. For a more complete discussion of MURDOCH itself, see [3].

MURDOCH is a general-purpose task-allocation system designed for use in dynamic environments in which many robots may come and go at any time. In order to support this kind of transience, we treat the entire collective as an anonymous pool of resources that can be applied to tasks that we want accomplished. MURDOCH solves the resultant resource allocation problem through the use of a variant of the Contract Net Protocol [11]. Essentially, when a task is

to be assigned, the task is put up for *auction*, and capable robots *bid* for it by stating their current fitness; the best-suited robot wins the auction and consequently executes the task.

With the goals of anonymity and scaling in mind, we have implemented the auction mechanism in MURDOCH using broadcast communication[4]; the task auctions themselves are *first-price sealed-bid* [7]. With regard to scaling, it is worth noting that the time spent by the robots in negotiation is negligible in comparison to the time required to execute tasks. The negotiation protocol is best understood by way of example so, in the interest of brevity, we will simultaneously explain the general protocol and its specific instantiation for use in `pusher-watcher`.

At the start, we have two robots with cameras, and a third with both a camera and a laser range-finder. We, as users, pose a `relocate-box` task to MURDOCH; this task is hierarchical and is in fact composed of one `watch-box` task that has two child `push-box` tasks. The `watch-box` task is put up for auction by an auctioneer (acting on behalf of the user) who broadcasts a *task announcement* that includes a list of *resources* required for the task: {`laser camera`} (the camera is used to sense the brightly-colored goal and the laser is used to sense the box's orientation). The one robot with those resources responds and, facing no competitors, claims the task and becomes our `watcher`. The `watcher` begins executing the `watch-box` task, which consists of: finding the goal, determining the angular error of the box, evaluating the control equations given in Section 3, and auctioning new `push-box` tasks.

Each `push-box` task requires only a single resource: {`camera`} (used to sense the brightly-colored box). The announcement is accompanied by a *metric* that potential pushers can use to score themselves as to their fitness for the task. In general, metrics can involve any arbitrary computation and take as input any part of the robot's state; in this case, the metric is a measure of how well-positioned the robot is for pushing on a certain end of the box. For example, when the task is to push on the right end of the box, the metric will reflect whether the box is offset to the left in the robot's visual field. Each candidate executes the metric and broadcasts its score back to the others, and so everyone immediately knows which robot was the winner (the robots are honest and tie-breaking mechanisms are built-in). The `watcher`, as auctioneer, awards the winner a time-limited task contract, then enters a monitoring phase. Left and right pushing tasks are allocated in pairs, parameterized with appropriate velocities, based on the orientation of the box.

We use time-limited contracts both because we cannot be sure of a robot's ability to complete a given task and because we may soon want to assign a different task. In

---

our box-pushing domain, each push-box task lasts 3 seconds. During those 3 seconds, the watcher monitors the progress of the task. As long as the box's orientation is unchanged the watcher simply renews the two pushing contracts. If the box's orientation changes significantly then the contracts are allowed to expire and new auctions are held for push-box tasks with re-computed velocities. The watcher also looks for robot failures that the faulty robot may not itself be able to detect. For example, if a single robot is given the task of pushing on the right end of the box and the box does not move even though the pushing robot reports that it is actually pushing, that robot has likely experienced some partial failure (e.g., its wheels are stuck). In this case, the watcher notes the failure and considers that robot to be ineligible for further tasks. It is important to note that time-limited contracts and progress monitoring are used at each level of the task hierarchy, including between the agent acting on behalf of the user and the watcher. Thus, should the watcher fail in some way, the top-level watch-box task would be automatically reassigned just as are the push-box tasks.

In a typical run of pusher-watcher, the watcher initially announces left and right push-box tasks with proper velocities, and lets them push until the box's orientation changes sufficiently to warrant different pushing velocities and thus new tasks. At that point, the current contracts are allowed to expire, and new ones are formed. This reactivity to world conditions is the feature that enables MURDOCH to dynamically reassign tasks in the face of robot failure. For example, when only a single robot is available, the watcher will actually try to allocate two pushing tasks as usual, but only one (the one with the higher velocity and thus higher priority) will be claimed. That single contract is renewed and the robot pushes on one end of the box until the orientation changes enough that it is more important to push the other end, at which point the robot will simply switch sides. When another robot is introduced, it will claim the next available pushing task and the two robots will work together at pushing the box.

5 Robot Platform

We implemented the pusher-watcher system on a group of ActivMedia Pioneer 2-DX mobile robots. These robots are non-holonomic, achieving locomotion through differential steering of two front drive wheels, with a passive caster in back. Many sensor configurations are possible with the Pioneer; for these experiments, each robot is equipped with a front sonar ring, a color camera, and a vision system that performs real-time color segmentation. The watcher robot is additionally equipped with a SICK laser range-finder that it uses to determine the relative orientation of the box being pushed.

Internally, each robot houses a Pentium-based computer running Linux, which executes its control program. Also on-board is Player[5] [4], a device server that handles low-level sensor and actuator control. Inter-robot communication is provided by way of wireless Ethernet; the topology is such that every robot on the network can communicate freely with every other robot.

As is the case with any choice of robots, our decision to implement pusher-watcher on this particular platform added extra constraints to the problem. We account for these constraints by implementing not the exact algorithm given in Section 3, but rather a suitable approximation. For example, although the algorithm can result in negative pushing velocities, the robots can only push, not pull, the box. Thus we bound the pushing velocities below by zero, which has the effect of increasing the minimum turning radius of the box (see Section 7). Further, it turns out that some pairs of pushing velocities, especially those with large differences, are extremely difficult to execute robustly in practice. This difficulty is due mostly to the nonholonomicity of our robots, which cannot move laterally; if a robot slips off the end of the box, it cannot re-acquire the box without performing a sort of "parallel-parking" maneuver, which is challenging in a dynamic environment. For this reason, we discretize the box's orientation space into bins (currently five) for which the resultant pushing velocities are practical.

6 Experiments

In order to evaluate the pusher-watcher system, we performed five sets of experiments on our group of Pioneer robots, as described below. Video footage of these experiments is available at: http://robotics.usc.edu/~agents/projects/markets-videos.html. During the experiments, we measured two quantities: success/failure and elapsed time. We define success as the situation in which the watcher declares that the task is terminated, and the center of the box is positioned within 0.5 meters of the target location; we do not specify a target orientation for the box. Conversely, a trial is a failure if either the watcher declares termination when the box is not close enough to the goal, or the box is rotated so far that the watcher can no longer perceive it using its laser range-finder (this threshold is approximately 70°). For ground-truth, we used an external metrology system consisting of multiple laser range-finders and beacons to track the positions of the box and robots throughout the experiments (trajectory plots in this paper come from that data).

As a control, Experiment Set 1 involved the simplest scenario. Two pusher robots had to move the box along a straight-line path for approximately 3 meters (90% of the length of our lab), and no failures were introduced.

---

[5]Player was developed at the USC Robotics Labs and is freely available at: http://robotics.usc.edu/player.

In Experiment Set 2, we tested the system's tolerance to an individual robot failure. The setup is the same as in Experiment Set 1, with two `pushers`, but, after they pushed the box approximately 1.2 meters, we simulated a robot failure by seizing one `pusher` and shutting it off. As a result, the remaining `pusher` was left to push the box by itself, alternating sides under the direction of the `watcher`. In Experiment Set 3, we tested MURDOCH's progress monitoring capability by introducing a partial robot fault. As with Experiment Set 2, two `pushers` began the task together; we then simulated a robot becoming stuck (e.g., in sand) by disabling its motor power. This failed robot was still in communication with the team and claimed to be fit for pushing tasks, but was immobile. In order to complete the task, the `watcher` had to detect the lack of progress on the part of the stuck robot and exclude it from future task offerings.

In Experiment Set 4, we tested the system's dynamic response by inducing both failure and recovery. We first let them both push approximately 0.6 meters, then removed one `pusher` to simulate complete failure. After the remaining `pusher` had single-handedly pushed the box another 1.2 meters or so, we re-introduced the failed `pusher`, at which point they had to finish the task together.

While Experiment Sets 1–4 showcase the ability of the `pusher-watcher` system to cooperatively move a box along a straight line, it is important to be able to follow more general paths. In Experiment Set 5, we tested the ability of the system to execute curved trajectories by placing the goal marker approximately 35° off to one side and 2.75 meters away from the box's initial location. In order to follow this non-straight path, the robots had to behave in a tightly coordinated fashion, making a series of rotational and translational adjustments.

7 Results

We performed 10 trials each from Experiments Sets 1–4. In the 40 trials, there were a total of four failures, one occurring in each set. Three failures were due to over-rotation of the box, and the fourth was due to premature termination on the part of the `watcher`, presumably because of sensor noise. With 36 successes in 40 trials, the two-sided 95% binomial confidence interval for the overall success rate of the system is: $p \in (0.76, 0.97)$. We also analyzed the time elapsed during the successful trials, as a measure of relative efficiency among the different experiments. The results are shown in Table 1. In Experiment Set 1, with no failure, the two `pushers` almost always executed the task in one continuous movement, yielding extremely similar (and short) completion times across trials. When one robot failed (Experiment Set 2) the completion time rose considerably because the remaining robot was left to push either side of the box in turn, which is rather inefficient compared to the two-robot cooperative case.

Experiment Set 3 (see Figure 3) took longer still because,

Set	Description	μ	σ
1	No failure (straight path)	31.22	0.44
2	Pusher failure	132.75	26.94
3	Partial pusher failure	260.89	37.79
4	Pusher failure & recovery	116.44	37.72

Table 1: *Mean (μ) and standard deviation (σ) of the elapsed time (in seconds) for the successful pushing trials in each of the four Experiment Sets.*
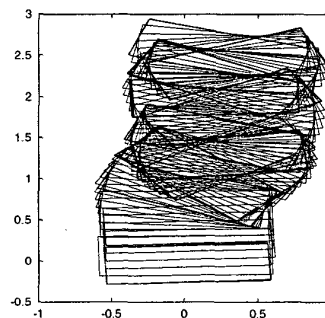

Figure 3: *Path of the box in an example trial from Experiment Set 3 (units are meters). The right* `pusher` *fails, leaving the other* `pusher` *to push either end of the box in turn.*

before the healthy `pusher` could take over the task on its own, the `watcher` had to recognize that the other robot was "stuck" and declare it unfit to participate. Exactly how long the `watcher` should wait is a system parameter that represents a trade-off between good dynamic response and the chance of falsely declaring a fault. The average completion time for Experiment Set 4 was less than for the other two failure modes, which suggests that the overhead required for coordination is outweighed by the improvement in performance from the robot's re-integration to the team. However, the large standard deviations preclude stronger comparisons. The magnitude of the standard deviations in the failure cases is explained intuitively by the complexity of the system; as this situation becomes more complicated, the exact behavior of the robots is less repeatable, due to the numerous interacting dynamic processes (e.g., variable torque output from motors, friction between the box and the floor).

We note that, in Experiment Set 4, after the failed robot recovered and was re-introduced, the `pushers` switched sides when appropriate, which was in 50% of the trials. The appropriateness of switching was determined by the configuration of the box and the remaining `pusher` at the time of re-introduction, and this configuration was in turn a result of the complex system dynamics mentioned above. However, the fact that the pushers automatically switched sides at the right times, with no detriment to the performance, demonstrates that our task-allocation system performs as specified.
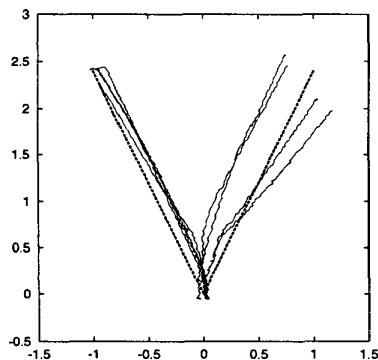
Figure 4: *Path of the box in trials from Experiment Set 5. Shown here is the trajectory of the center of the box for four successful trials in either direction (units are meters). For comparison, the dashed lines represent the "ideal" paths.*

In addition to verifying the *validity* of the task assignments made by MURDOCH in the pusher-watcher system, we also want to know their *quality*. We evaluate the pusher-watcher system in this respect by comparing the measured trajectories of the box to an "ideal" trajectory. For this comparison, we define the ideal trajectory as the one which causes the center of the box to follow the shortest path to the goal. This path is simply the straight line from the initial location of the box to the goal. However, this trajectory is unrealistic because it would require that the box be rotated in place at the start, a feat of which the physical robots are not capable. Shown in Figure 4 are measured and ideal box trajectories for trials in Experiment Set 5. We can see from this data that the pusher-watcher system generates efficient paths that are close to the (unachievable) ideal.

8 Conclusion

We have described the design and implementation of pusher-watcher, a novel distributed algorithm for box-pushing by teams of mobile robots. Using the auction-based task-allocation facilities provided by MURDOCH, pusher-watcher is tolerant to robot failures, and efficient in its use of resources. We have demonstrated this system through a series of experiments on a group of Pioneer mobile robots. The results from these experiments are encouraging and represent, to our knowledge, the most sophisticated and successful multi-robot box-pushing system validated on physical robots.

We are currently exploring other directions of this work, including different control laws, better task metrics, and more robots. Specifically we plan to investigate the use of multiple watchers, which would overcome the limited field of view provided by a single watcher, as well as provide more robust behavior in the face of watcher failures.

In addition, more complex trajectories (including the negotiation of obstacle fields) could be accommodated through the use of multiple sub-goals as via-points. We plan to use this system as part of a larger resource transportation task to move large objects through complex environments such as building corridors.

Acknowledgments

References

[1] Bruce Donald, Jim Jennings, and Daniela Rus. Information invariants for distributed manipulation. *The Intl. J. of Robotics Research*, 16(5):673–702, October 1997.

[2] Rosemary Emery and Tucker Balch. Behavior-based control of a non-holonomic robot in pushing tasks. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2381–2388, Seoul, South Korea, 2001.

[3] Brian P. Gerkey and Maja J Matarić. Principled communication for dynamic multi-robot task allocation. In D. Rus and S. Singh, editors, *Experimental Robotics VII, LNCIS 271*, pages 353–362. Springer-Verlag Berlin Heidelberg, 2001.

[4] Brian P. Gerkey, Richard T. Vaughan, Kasper Støy, Andrew Howard, et al. Most Valuable Player: A Robot Device Server for Distributed Control. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1226–1231, Wailea, Hawaii, October 2001.

[5] C. Ronald Kube and Hong Zhang. The use of perceptual cues in multi-robot box-pushing. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2085–2090, Minneapolis, Minnesota, April 1996.

[6] Maja J Matarić, Martin Nilsson, and Kristian T. Simsarian. Cooperative multi-robot box-pushing. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 556–561, Pittsburgh, Pennsylvania, August 1995.

[7] R. Preston McAfee and John McMillan. Auctions and Bidding. *J. of Economic Literature*, 25(2):699–738, June 1987.

[8] Fabrice R. Noreils. Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment. *The Intl. J. of Robotics Research*, 12(1):79–98, 1993.

[9] Lynne E. Parker. ALLIANCE: An Architecture for Fault Tolerant, Cooperative Control of Heterogeneous Mobile Robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 776–783, Munich, Germany, September 1994.

[10] Lynne E. Parker. ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.

[11] Reid G. Smith. The Contract Net Protocol. *IEEE Transactions on Computers*, 29(12):1104–1113, December 1980.

[12] J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor, et al. Cooperative Localization and Control for Multi-Robot Manipulation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 631–636, Wailea, Hawaii, October 2001.

[13] Barry Brian Werger and Maja J Matarić. Broadcast of Local Eligibilty for Multi-Target Observation. In Lynne E. Parker, George Bekey, and Jacob Barhen, editors, *Distributed Autonomous Robotic Systems 4*, pages 347–356. Springer-Verlag, 2000.