Proceedings, Naval Research Laboratory Workshop on Multi-Robot Systems, Washington, DC, March 17 - 19, 2003, 65-76

COOPERATIVE RELATIVE LOCALIZATION FOR MOBILE ROBOT TEAMS: AN EGO-CENTRIC APPROACH

Andrew Howard, Maja J Matarić and Gaurav S. Sukhatme
Robotics Research Laboratory, Computer Science Department,
University of Southern California,
Los Angeles, California, 90089-0781, U.S.A
ahoward@usc.edu, mataric@usc.edu, gaurav@usc.edu

Abstract This paper describes a cooperative *relative* localization method for mobile robot teams. That is, it describes a method whereby each robot may determine the pose of every other robot in the team, relative to itself. This method does not require GPS, landmarks, or any kind of environment model. Instead, robots make direct measurements of the relative pose of nearby robots, and broadcast this information to the team as a whole; each robot processes this information to generate ego-centric estimates of the pose of other robots, including those robots that they cannot observe directly. The method makes use of a Bayesian formalism and particle filter implementation, and is, as a result, very robust. The system described in this paper will both self-initialize (i.e., it does not require a priori pose estimates) and self-correct (it can recover from tracking failures). The method is well suited to applications involving unstructured, unknown or non-stationary environments.

Keywords: Localization, multi-robot systems, robot teams.

1. Introduction

Localization is a key capability for both individual mobile robots and for mobile robot *teams*. For robot teams, however, one must distinguish between two kinds of localization: *absolute* localization, in which each robot attempts to determine its pose with respect to some external global coordinate system, and *relative localization*, in which each robot attempts to determine the pose of every other robot in the team, *relative to itself*. For many team-oriented behaviors, it is this latter kind of localization that is most important. Consider, for example, a team of robots executing a formation behavior: these robots need not know their latitude and longitude, but *must* know the relative pose of their neighbors.

Naturally, given a set of absolute pose estimates for the robots, one can always derive relative poses. Absolute localization, using either GPS or model-based methods, is a well studied problem (Leonard and Durrant-Whyte, 1991; Simmons and Koenig, 1995; Fox et al., 1999), and will not be treated here. Instead, we described an approach in which robots make relative pose estimates *directly*, through the observation of other robots. We make the following assumptions. First, we assume that each robot is equipped with a *robot sensor* that allows it to measure the relative pose and identity of nearby robots; robot sensors can be readily constructed using cameras and/or scanning laser range finders in combination with coded fiducials. Second, we assume that each robot in the team is equipped with a *motion sensor* with which it can measure changes in its own pose; odometry or inertial measurement units are typically used for this purpose. Finally, we assume that robots are able to communicate using some broadcast medium (e.g., a wireless network). For simplicity, we assume that communication is both reliable and complete, but note that this is not essential to the method.

The basic approach is as follows. Consider a single robot taken from a team of n robots; this robot attempts to estimate the pose of every other robot in the team *relative to itself.* To do this, it maintains a set of $n-1$ probability distributions, each of which describes the pose of one other robot. These pose estimates are described in an *ego-centric* coordinate system that is fixed to the body of the original robot (i.e., the robot is always at the origin of its own coordinate system). These pose estimates are updated in response to observations obtained from both the robot's own sensors, and from the sensors of other robots (robots are required to broadcast their observations to the team). There are five basic types of observations that must be considered: from the motion sensors we may observe either that *this* robot has moved, or that some *other* robot has moved; from the robot sensors we may observe that this robot has observed another robot, that another robot has observed this robot, or that two other robots have observed each other. The update rules used for the first four types of observations are fairly straight-forward, and are similar to those used in single-robot Markov localization techniques (Fox et al., 1999). The update rules for the last type of observation, however, are somewhat more complicated. This type of observation defines a relationship between two probability distributions, and one must pay careful attention to the dependencies that exist between these two distributions. In Section 3.3, we describe an approximate method for monitoring such dependencies using the notion of a *dependency tree.*

The ego-centric approach described in this paper has a number of attractive features. It does not require the use of external landmarks or environment models, and is therefore well suited for use in unknown, unstructured and non-stationary environments. It is highly decentralized (each robot maintains its own set of $n-1$ distributions), yet requires minimal communication (robots need only transmit their observations). In addition, our particular implementation, which makes use of both particle filters and mixture sampling (Thrun et al., 2001), is extremely robust. It is able to both self-initialize and self-correct, and can thus solve both the weak and the strong form of the kidnapped robot problem. The principal drawback of the ego-centric approach is its computational cost; particle filters are relatively expensive, and each robot in a team of n robots must maintain $n-1$ separate filters. Our empirical results are promising, however: for a team of four mobile robots, we can easily run the system in real-time using conventional laptop computers.

The remainder of this paper sketches the basic formalism underpinning the ego-centric approach and describes its implementation using particle filters. We also present experimental results obtained using a team of four mobile robots.

2. Related Work

Localization is an extremely well studied problem in mobile robotics. The vast majority of this research, however, has concentrated on two problems: localizing a single robot using an a priori map of the environment (Leonard and Durrant-Whyte, 1991; Simmons and Koenig, 1995; Fox et al., 1999), or localizing a single robot while simultaneously building a map (Thrun et al., 1998; Lu and Milios, 1997; Yamauchi et al., 1998; Golfarelli et al., 1998; **?**; Dissanayake et al., 2001). Recently, some authors have also considered the related problem of map building with multiple robots (Thrun, 2001). All of these authors make use of statistical or probabilistic techniques; the common tools of choice are Kalman filters, maximum likelihood estimation, expectation maximization, and Markov models.

Among those who have considered the specific problem of cooperative localization are (Roumeliotis and Bekey, 2000) and (Fox et al., 2000). Roumeliotis and Bekey present an approach to multi-robot localization in which sensor data from a heterogeneous collection of robots are combined through a single Kalman filter to estimate the pose of each robot in the team. They then show how this centralized Kalman filter can be broken down into n separate Kalman filters (one for each robot) to allow for distributed processing. In a somewhat similar vein, Fox *et al.*

describe an approach to multi-robot localization in which each robot maintains a probability distribution describing its own pose (based on odometry and environment sensing), but is able to refine this distribution through the observation of other robots. This approach extends earlier work on single-robot Markov/Monte-Carlo localization techniques (Fox et al., 1999). Note that both of these authors are principally concerned with determining the *absolute* pose of robots (with or without the use of external landmarks).

A number of authors (Kurazume and Hirose, 2000; Rekleitis et al., 1997) have considered the problem of cooperative localization from a somewhat different perspective. These authors describe *active* approaches to localization, in which team members carefully coordinate their activities in order to reduce cumulative odometric errors. While our approach does not require such explicit cooperation on the part of robots, the accuracy of localization can certainly be improved by the adoption of such strategies.

3. Formalism

Consider a single robot drawn from a team of n robots; we will refer to this robot as *self*. Let x_i denote the current pose of some other robot i relative to self. Our aim is to compute the probability distribution $p(x_i)$ for each such robot in the team, based on observations made both by self and by other robots.

As previously indicated, we assume that each robot is equipped with two sensors: a *robot sensor* that allows it to measure the relative pose and identity of nearby robots, and a *motion sensor* that allows it to measure changes in its own pose. Together, these two sensors give rise to five distinct types of observations. For the motion sensor there are observations made by self, and observations made by another robot. For the robot sensor there are observations in which self observes another robot; observations in which another robot observes self; and observations in which another robot observes a third robot. We refer to this last class of observations as *transitive*, since there is an intermediary between self and the robot being observed. For each kind of observation, we require a different update rule for the probability distributions $p(x_i)$. Since transitive observations in fact require *two* update rules (to be used in different situations, as we will show) there are six update rules in total. In the sections that follow, we consider the qualitative form for each these rules; for a more mathematical treatment, see (Howard et al., 2003).

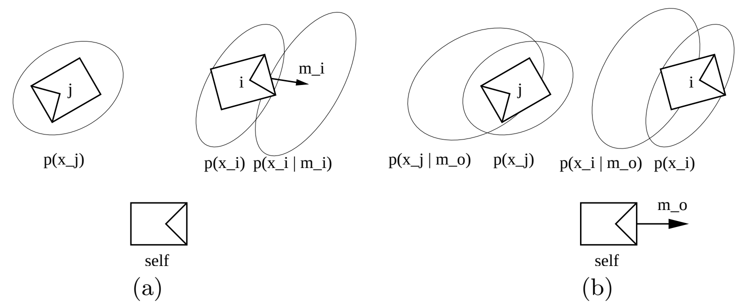


Figure 1. Update rules for motion observations. The ellipses denote the shape of the robot pose distributions before and after the update is applied. (a) A motion observation m_i made by robot i. (b) A ego-motion observation m_o made by *self*; note that all distributions must be updated in response to an ego-motion observation.

3.1 Motion Observations

Let m_i denote a motion observation made by robot i; that is, m_i describes the measured change in pose of some robot i between times t and t'. This observation affects only the distribution $p(x_i)$, which must be both translated (to account for the robot's motion) and 'blurred' (to account for the uncertainty in that motion). This situation is illustrated in Figure 1(a).

Let m_o denote a motion observation made by *self*. Since self is, by definition, at the origin of the coordinate system, this observation is used to update the pose distributions $p(x_i)$ for all *other* robots. That is, all distributions must be both translated (to account for self's motion) and 'blurred' (to account for the uncertainty in that motion). This situation is illustrated in Figure 1(b). Note that the translation is in the opposite direction to self's motion.

3.2 Robot Observations (Non-Transitive)

Let r_{io} denote an observation made by self of some robot i; that is, r_{io} describes the measured pose of robot i relative to self. This observation affects only the distribution $p(x_i)$, which can be updated through the direct application of Bayes' Law. In general, such updates will 'sharpen' the distribution, reducing the uncertainty in the pose estimate. This situation is illustrated in Figure 2(a).

Let r_{oi} denote an observation made by robot i of self; i.e., r_{oi} describes the measured pose of self relative to robot i. Naturally, it makes no sense to update the pose estimate for self (which is, by definition, at the origin of the coordinate system); instead, such measurements are used

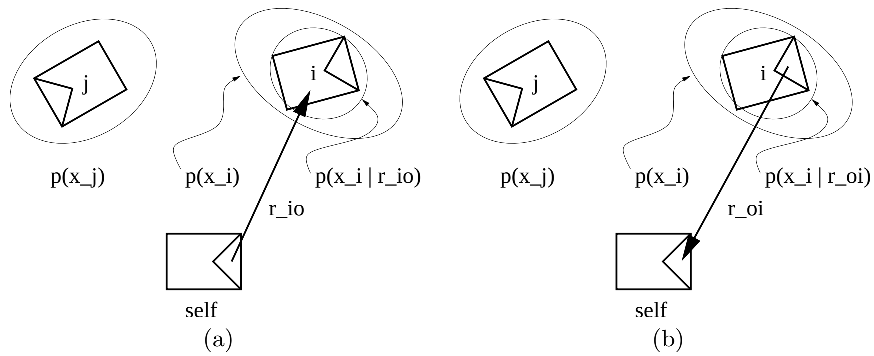


Figure 2. Update rules for non-transitive robot observations. The ellipses denote the shape of the robot pose distributions before and after the update is applied. (a) Self observes robot i. (b) Robot i observes self.

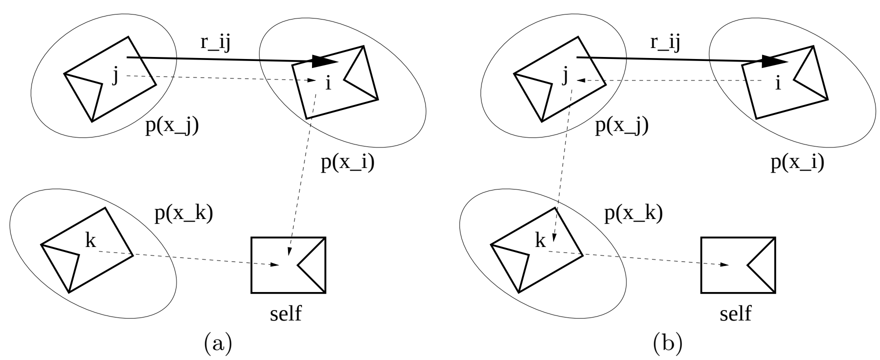


Figure 3. Update rules for transitive robot observations. The ellipses denote the shape of the robot pose distributions before the update is applied; the dashed lines denote the dependency tree. (a) Robot j observes robot i, and since i is an ancestor of j we update $p(x_j)$. (b) Robot j observes robot i, and since j is an ancestor of i, we update $p(x_i)$.

to update the pose estimate for robot i. This situation is illustrated in Figure 2(b). Once again, the update can be performed using Bayes' Law; note, however, that the sense of the observation must be inverted.

3.3 Robot Observations (Transitive)

Transitive robot observations, i.e., observations in which some robot i observes another robot j (with neither robot being *self*), must be treated with some care. In particular, we must pay attention to *dependencies* that may exist between the distributions $p(x_i)$ and $p(x_j)$. Consider, for example, a scenario in which robot i observes robot j, which then observes robot i. If we were to use the first observation to update the distribution $p(x_j)$, and subsequently use the second observation to update $p(x_i)$, we would be guilty of double-counting: the first observation would effectively be used to update $p(x_i)$ not once, but twice. This kind of circular reasoning can easily lead to over-convergence, with the pose estimates for a pair of robots i and j quickly converging to some precise but inaccurate value.

In order to avoid such circular reasoning, we maintain a *dependency tree* for all $n-1$ distributions. In this tree, each distribution has exactly one *parent* distribution, and zero or more *child* distributions. A distribution cannot be used to update its ancestors, but may be used to update its descendants; whenever a distribution is used in this manner, it becomes the new parent of the distribution being updated (the topology of the graph may therefore change as new observation are added). Given some transitive observation r_{ij}, the dependency tree is used to decide which of the two distributions $p(x_i)$ or $p(x_j)$ will be updated. If i is an ancestor of j, we update j; if j is an ancestor of i, we update i. Figure 3 illustrates these update rules. In the case that neither distribution is an ancestor of the other, we have a free choice as to which distribution should be updated; in this case, we update distribution with the greatest variance.

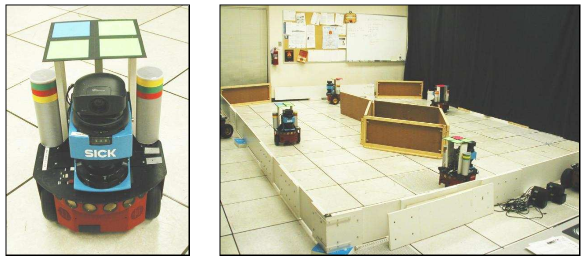
While the dependency tree does prevent the most obvious circular reasoning, it remains an incomplete and approximate solution. The dependency tree assumes that distributions are dependent *only* on the distribution that was last used to update them, and are therefore *independent of all other distributions*. Thus, it is still possible to construct situations in which circular reasoning occurs. Fundamentally, the dependency problem arises from the fact that we use *separate* probability distributions to represent the pose of each robot. An alternative approach would be to maintain the *joint* probability distribution $p(x_i, x_j)$ for all i, j pairs, and project these distributions when necessary to generate $p(x_i)$ and $p(x_j)$. This approach eliminates the dependency problem, and has the added advantage of making better use of transitive observations (the dependency tree approach effectively discards much of the information carried in such observations). On the other hand, this latter approach requires that we maintain $(n-1)^2$ joint distributions, each defined over a high dimensional space. Thus, this approach may become intractable as team size increases. In comparison, the dependency tree appears to be a cheap and effective solution.

4. Implementation

The formalism outlined in the previous section has only three inputs: a motion sensor model, a robot sensor model and a stream of observations. The sensor models must be known a priori, and can be determined either analytically or empirically. Note that these models capture the properties of the robots and their sensors; no environment model is required. Observations are made by all robots on the team, and are shared between robots using UDP broadcast sockets (recall that every robot on the team is trying to estimate the pose of every *other* robot on the team). The bandwidth required for this is of the order of a few hundred bytes per second per robot, and hence represents a very modest communications overhead.

The most difficult aspect of the implementation is, of course, maintaining the (non-parametric) probability distributions $p(x_i)$, which we model using *particle filters*. Consider once again the individual robot *self*. Self maintains a set of $n - 1$ independent particle filters, each of which represents the relative pose distribution $p(x_i)$ for another robot. Each filter contains a relatively large number of *samples*, each of which represents one possible pose for that robot. Samples have an associated *weight*, corresponding to the probability that this sample reflects the true robot pose. For each observation, one or more of the particle filters must be updated; loosely speaking, motion observations will translate the samples, while robot observations will alter their weight.

The full details of particle filter implementation are relatively complex, and will not be described here. One particular feature of the implementation should be highlighed, however: the use of a *mixture sampling algorithm* (Thrun et al., 2001) to update distributions. This algorithm creates two distributions from each robot observation: the first is generated by applying the update rules described in Sections 3.2 and 3.3 to an existing distribution, the second is generated by considering the new observation in isolation, and creating a distribution that contains only those poses that are compatible with this observation. The final, updated, distribution is a mixture of the two. Use of this mixture sampling algorithm greatly increases the robustness of the implementation. Although samples will be drawn predominantly from the first distribution, the presence of samples from the second distribution allows filters both to self-initialize and to self-correct. Thus, for example, in the absence of any a priori pose information, the first few observations will effectively 'seed' the filter with a set of drawn samples from the second distribution. Later, if for some reason the filter ceases to track the true



(a) (b)

Figure 4. (a) One of the robots used in the experiment. The robot is equipped with a scanning laser range-finder, a pan-tilt-zoom camera, a pair of retro-reflective and color-coded fiducials, and a second color-coded fiducial for use with the overhead tracking/ground-truth system. (b) The experimental environment.

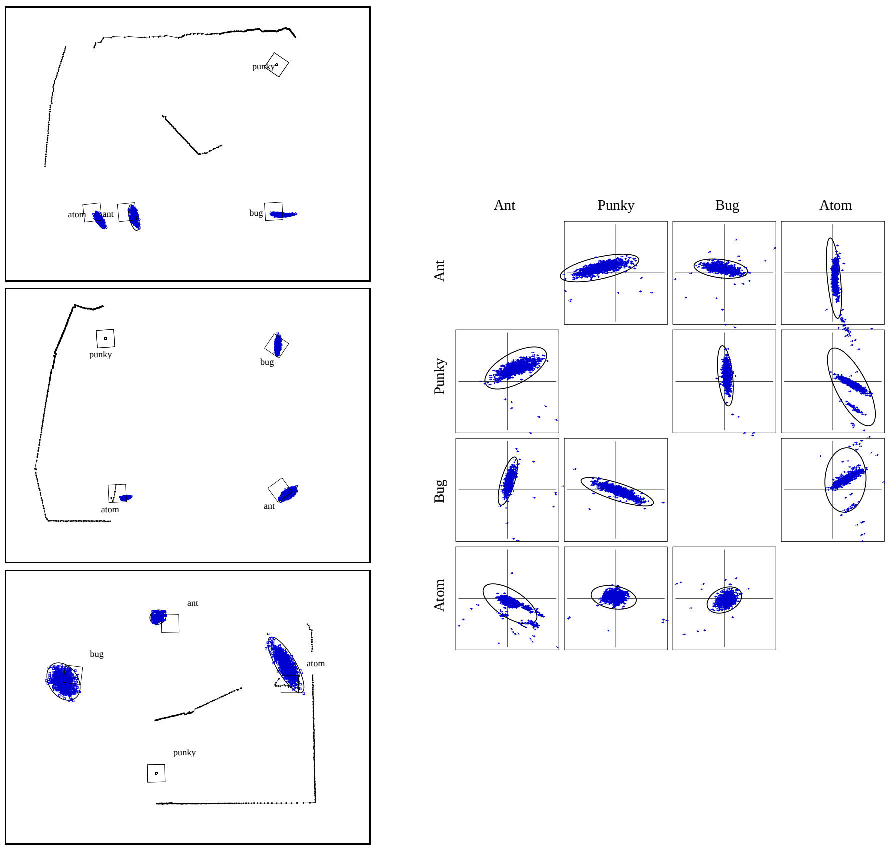
robot pose, new samples added by the second distribution will eventually re-initialize the filter to the correct value.

5. Experiments

We have conducted a number of experiments aimed at determining the accuracy of the ego-centric approach and the robustness of our particular implementation. We describe one such experiment here.

The experiment was performed using four Pioneer 2DX mobile robots running the *Player* robot device server (Gerkey et al., 2001). Each robot was equipped with a SICK scanning laser range-finder and a Sony pan-tilt-zoom camera (see Figure 4(a)); the laser and camera are used in tandem to detect and identify fiducials placed on the robots (the fiducials are both retro-reflective and color-coded). With appropriately placed fiducials, the range, bearing, orientation and identity of each robot can be uniquely determined. Ground-truth data was generated by the *Mezzanine* tracking package, which uses multiple overhead cameras to track a second set of color-coded fiducials placed on top of the robots. The tracking system is accurate to about ± 10 cm and $\pm 2°$. Note that this system was used solely to generate comparative ground-truth data; no information from the tracking system was provided to the robots. The environment for the experiment was a simple pen measuring approximately 8 by 6 meters, containing various obstacles (Figure 4(b)). The robots executed a simple wall-following behavior in this pen, performing 6 complete circuits without pausing, and making only intermittent observations of one another.

Figure 5(a) shows a set of 'snap-shots' taken from one of the robots (Punky) during the course of the experiment. Each snap-shot shows the probability distributions generated by the robot Punky for each of



(a) (b)

Figure 5. (a) Snap-shots showing the pose estimates generated by one of the robots (Punky) for each of the other robots. The particle clouds show the pose estimates, the ellipses show the 3σ interval, and the squares denote the true robot poses (as determined by the overhead tracking system). (b) A snap-shot taken at time $t = 50$s summarizing the difference between the true and estimated poses. Each row contains the estimates generated *by* one of the robots; each column contains the pose estimates generated *for* one of the robots. The estimates are shown relative to the true pose. Each plot is 2 m on a side.

the robots Bug, Ant and Atom; i.e., these distributions correspond to Punky's estimates for the relative pose of each of the other robots. The true pose of each of the robots is also shown, and the distributions have been plotted in the ground-truth coordinate system to simplify analysis. There are a number of features in this figure that should be noted. First, all of the pose distributions have been correctly initialized, despite the fact that no a priori pose information was provided; the distributions in these figures are generated entirely by observations made during the course of the experiment. Second, Punky is able to estimate the pose of every other robot, including those robots that it has never observed directly. Thus, for example, Punky is able to estimate the pose of Ant, despite the fact that it has never seen nor been seen by this robot.

Figure 5(b) shows a somewhat different view of the data. This is a snap-shot taken at time $t = 50$ s; each row shows the set of estimates generated *by* one of the robots; each column shows the set of estimates generated *for* one of the robots. Furthermore, each cell shows the *difference* between the pose estimate and the true relative pose. Thus, for example, the top-right cell shows the difference between Ant's estimate of Atom's relative pose and the true relative pose of these two robots. The ellipses in Figure 5 show the 3σ intervals for each of the distributions; if our sensors models are accurate, and if the distributions are Gaussian, this ellipse should capture the origin 99.9% of the time. The actual figure for this experiment is 98%, which is acceptable for most practical purposes.

6. Conclusion and Further Work

This paper describes an method whereby each member in a mobile robot team may determine the pose of every other robot relative to itself. The approach is entirely decentralized and requires minimal communication between the robots; in addition, the experimental results presented in Section 5, while preliminary, suggest that this method is both accurate and robust.

The ego-centric approach described in this paper also admits a number of interesting extensions. For example, while the approach does not *require* the use of GPS, it could easily be extended to exploit GPS information when it is available. As a consequence, one can envisage a situation in which only some of the robots in a team are able to recieve a GPS signal, yet all of the robots are able to compute their GPS location. This particular capability may be of great interest for teams operating in mixed urban indoor/outdoor environments.

Acknowledgments

This work is supported in part by the DARPA MARS Program, grant DABT63-99-1-0015, and ONR DURIP grant N00014-00-1-0638.

References

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

Fox, D., Burgard, W., Kruppa, H., and Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots, Special Issue on Heterogeneous Multi-Robot Systems*, 8(3):325–344.

Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhatme, G. S., and Matarić, M. J. (2001). Most valuable player: A robot device server for distributed control. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS01)*, pages 1226–1231, Wailea, Hawaii.

Golfarelli, M., Maio, D., and Rizzi, S. (1998). Elastic correction of dead reckoning errors in map building. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 905–911, Victoria, Canada.

Howard, A., Matarić, M. J., and Sukhatme, G. S. (2003). Putting the 'I' in 'Team': An ego-centric approach to cooperative localization. Submitted to *Proc. of the IEEE International Conference on Robotics an Automation (ICRA03)*, Taipei, Taiwan.

Kurazume, R. and Hirose, S. (2000). An experimental study of a cooperative positioning system. *Autonomous Robots*, 8(1):43–52.

Leonard, J. J. and Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382.

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

Rekleitis, I. M., Dudek, G., and Milios, E. (1997). Multi-robot exploration of an unknown environment: efficiently reducing the odometry error. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1340–1345.

Roumeliotis, S. I. and Bekey, G. A. (2000). Collective localization: a distributed Kalman filter approach. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2958–2965, San Francisco, U.S.A.

Simmons, R. and Koenig, S. (1995). Probabilistic navigation in partially observable environments. In *Proc. of International Joint Conference on Artificial Intelligence*, volume 2, pages 1080–1087.

Thrun, S. (2001). A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363.

Thrun, S., Fox, D., and Burgard, W. (1998). A probabilistic approach to concurrent mapping and localisation for mobile robots. *Machine Learning*, 31(5):29–55. Joint issue with Autonomous Robots.

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

Yamauchi, B., Shultz, A., and Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. In *Proc. of the 1998 IEEE/RSJ International Conference on Robotics and Automation*, volume 4, pages 3175–3720, San Francisco, U.S.A.