Coordinating Aerial Robots and Sensor Networks for Localization and Navigation

Peter Corke[1], Ron Peterson[2], and Daniela Rus[3]

[1] CSIRO ICTC Centre,
Brisbane, 4069 Australia
peter.corke@csiro.au
[2] Dartmouth Computer Science Department,
Hanover, NH 03755 USA
rapjr@cs.dartmouth.edu
[3] Computer Science and Artificial Intelligence Lab
Cambridge, MA 02139
rus@csail.mit.edu

We consider multi-robot systems that include sensor nodes and aerial or ground robots networked together. We describe two cooperative algorithms that allow robots and sensors to enhance each other's performance. In the first algorithm, an aerial robot assists the localization of the sensors. In the second algorithm, a localized sensor network controls the navigation of an aerial robot. We present physical experiments with an flying robot and a large Mica Mote sensor network.

1 Introduction

We wish to develop distributed networks of sensors and robots that perceive their environment and respond to it, anticipating information needed by the network and by users of the network, repositioning and organizing themselves to best acquire and deliver that information. These networks, thousands of small sensors, equipped with limited memory, sensing, communication, and actuation capabilities will autonomously organize themselves and move to track a source and convey information about its location to a human user, and to the rest of the system.

In this paper we discuss the cooperation between a ground sensor-network and a flying robot. We assume that the flying robot is connected by point-to-point communication with a ground sensor network. The nodes of the sensor network are simple and they support local sensing, communication, and computation. The communication range of all nodes is limited, but the resulting mobile sensor network supports multi-hop messaging. The flying robot makes the sensor network localization easy by providing access to GPS data to all nodes. In turn, the sensor network helps the navigation of the flying robot by providing information outside the robot's imme-

diate sensor range. In our previous work [2] we discuss the details of robot-assisted localization. In this paper we summarize robot-assisted localization, discuss why it is hard, and present a new sensor-assisted robot guidance algorithm. We also describe new experimental results with an integrated testbed for robot-assisted localization and sensor-assisted guidance.

2 Robot-assisted localization

Ground sensor networks are usually deployed on-demand, so that the location of the sensors may not always be presettable by the deployment system. Once on the ground the sensors acquire location information autonomously and form a distributed system. The individual sensor nodes are too simple to include complex sensors such as GPS, or to support complex information processing tasks estimating location from range measurements.

The node localization problem has been previously discussed by others and usually requires estimates of inter-node distance, a difficult problem. Simić and Sastry [8] present a distributed algorithm that localizes a field of nodes in the case where a fraction of nodes are already localized. Bulusu etal. [1] propose a localization method that uses fixed beacons with known position. Galystyan etal. [3] described a constraint-based method whereby an individual node refines its position estimate based on location broadcasts from a moving agent. We wish to address the sensor localization problem in a uniform and localized way, without relying on beacons or pre-localized nodes, while minimizing the number of broadcasts required.

In [2] we introduced the idea of robot-assisted localization, an approach to localization that is orthogonal to this previous work, does not require inter-node communication, and is suitable for sensor networks deployed outdoors. For a large sensor network the location requirement could be limiting since it would be impractical (for reasons of cost and power consumption) for each node to have GPS capability. However, a mobile aerial robot equipped with a GPS system can assist the sensors to localize. The aerial robot sweeps across the area of the sensor network, for example along a random path or a path defining a grid, broadcasting GPS coordinates. The sensors process all broadcasts they hear and estimate their location. If the mobile node beams messages containing its position $p_i = (x_i, y_i)$ any sensors receive the message with signal strength s_i, a simple averaging procedure can estimate a sensor's location as the centroid of the set of GPS locations heard over time. Other methods discussed in [2] include taking just the strongest received signal, a signal strength weighted mean, a median, a set intersection approach as suggested by Galystyan etal. [3]. The latter requires a parameter which is the notional reception range of the radio, assumed to be circular. Note that algorithms **mean**, **wmean** and **median** can be modified so that the estimate is only updated when $s_i > s_{min}$ which artificially reduces the size of the radio communications region.

2.1 Challenges with Distributed Localization

In the robot-assisted localization algorithm, the robot regularly broadcasts its location. When within the reception range of the sensor, these broadcasts provide input to the localization algorithm. The reception range is not symmetrical due to the lobe shape of both the transmitting and receiving radios involved, terrain, etc. Since the asymmetry depends on the relative orientation of both antennas it will vary from encounter to encounter, which highlights two problems.

1. The asymmetry is not known apriori, so the best we can do is to approximate the center of the radio reception range, i.e., assume the sensor is at the center of the radio reception range. Node 7 in Figure 3(a) shows an extreme case of directional reception.
2. With relatively few measurements occurring within the reception range the estimate of centroid will be biased.

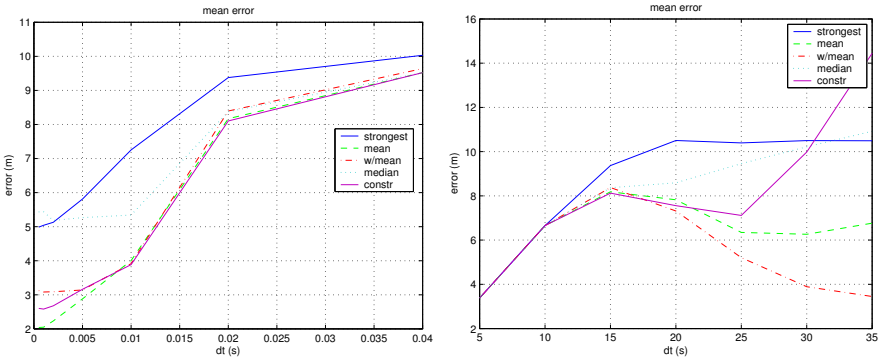
The first problem is not solvable given current radios Multiple encounters at different relative antenna orientations might provide some relief, but would increase the time and cost of any post-deployment localization phase.

There are ways to improve the second problem however:

1. Increase the rate at which position broadcasts are sent, giving more samples within the reception range, and improving the estimate of the centroid.
2. Increase the size of the reception range in order to acquire more samples. One way to do this would be to relay messages between close neighbors, perhaps based on a hop-count estimate of distance. A disadvantage of this method is that the asymmetry problem is likely to be exacerbated.
3. Decrease the size of the reception range, perhaps combined with improvement #1, so that those broadcasts that are received originate very close to the sensor.

In early simulation studies we observed that the localization result is strongly dependent on the path of the robot with respect to the deployed nodes. To sidestep this dependence while testing observations (1–3) above our simulation uses a fixed serpentine robot path and 100 sensors deployed randomly with a uniform distribution in a square region 100×100m (mean inter-node spacing is 17m). The robot starts at the origin in the lower-left corner, moves 100m to the right, up 20m, 100m to the left, then up another 20m and repeats the cycle. The total time to execute this path is 1 unit, and we investigate the effect of changing the broadcast interval. The radio propagation model assumes that signal strength decreases with distance and becomes zero at the maximum distance parameter which we also vary.

For each set of simulation parameters, such as radio range or position broadcast rate, we compute mean and maximum localization error. We repeat the experiment 100 times, and compute second-order statistics. For each experiment, for each node, we run the 5 localization algorithms previously described. Figure 1 shows some of the results.



(a) (b)

Fig. 1. Mean localization error from the Monte Carlo study using the five methods of [2]. (a) Effect of varying the broadcast interval (transmit range = 15m). (b) Effect of varying the transmission radius (dt=0.02).

We observe that as the number of broadcasts increases (ie. broadcasts are closer together) the localization error decreases and reaches a plateau at around 5m or better. The method **strongest** performs least well while **constr** performs best. For a given number of broadcasts, 50, along the path we investigate the performance of the methods for varying transmit radius. We see that the method **constr**, previously a strong performer, breaks down when the actual and assumed transmit radii are not equal. The best performer in this test is **wmean**, though **mean** also behaves well.

3 Sensor-assisted navigation

A localized set of sensors can facilitate the aerial robot's navigation by encoding path information which provides the robot with point-by-point navigation directions using networking. The path can be communicated to the robot by the ground sensors. The sensor network can employ mapping algorithms such as those described in [6] to compute adaptive, time-varying paths to events. One application of this approach is in the area of monitoring and surveillance, where the sensor network may detect something that requires further investigation with a more complex sensor, say with the camera on board of the flying robot.

Suppose a path is stored in the sensor field. Sensor-assisted navigation for the flying robot has two phases: firstly getting to where the path starts, and secondly being guided along the path. In some situations the first phase may not be needed (e.g., the path may always be computed to include the known location of the robot or the robot could always be told where the start of the path is.)

One important goal in this first phase is to avoid flooding the entire network with messages in an attempt to discover location. Algorithm 1 summarizes a method for guiding the robot to the path. For example, for the robot to find the path, first one (or

Algorithm 1 The FindPath algorithm to get the robot to the start of the path.

The sensor does this to announce the location of a path start to the robot.

if Incoming message is a *PathMessage* AND this sensor is at the start of a path **then**
  Broadcast *FindRobotMessage* with 0 degree heading to MAXRANGE distance
  Broadcast *FindRobotMessage* with 120 degree heading to MAXRANGE distance
  Broadcast *FindRobotMessage* with 240 degree heading to MAXRANGE distance
else if Incoming message is a *FindPathMessage* **then**
  if This sensor is storing a path start location **then**
    Broadcast a *PathStartMessage*
else if Incoming message is a *PathStartMessage* **then**
  Compute *distance* to vector from path start to robot.
  if *distance* $<$ *PathMessage.PathWidth* **then**
    // Forward message towards the robot.
    Rebroadcast *PathStartMessage*

The robot does this to find the start of the path.

while forever **do**
  // Seek the path start
  Broadcast *FindPathMessage* with 0 degree heading to MAXRANGE distance
  Broadcast *FindPathMessage* with 120 degree heading to MAXRANGE distance
  Broadcast *FindPathMessage* with 240 degree heading to MAXRANGE distance
  if A *PathStartMessage* is received **then**
    Store location of start of path.
    Head for start of path.
    break

all) of the sensors that know they are near the start of the path send out three messages that contain the location of the start of the path. The messages also each contain a heading, set 120 degrees apart[4], a width for the vector they will travel along, and a maximum range beyond which they are not intended to travel. The messages are forwarded out to that range in each of the three directions. The sensors that forward the messages store the location of the start of the path.

The robot at some later time sends out the same sort of messages in three directions. If the robot and path start are in range of each other's messages, the message paths will cross (due to using a 120 degree dispersal angle.) The sensor(s) at the crossing will have a stored location for the start of the path and a location for the robot and can send a directional message (perhaps with a gradually increasing width since the robot may have moved slightly) back to the robot telling it where the start of the path is. In this way only the sensors along specific lines extending to a maximum range carry messages instead of the entire network.

[4]Other patterns of radiation (a star pattern of 72 degrees) might increase the likelihood of intercepts occurring, though they also increase the number of sensors involved.

Algorithm 2 The QueryPath algorithm for robot guidance.

```
while forever do
  // Seek path information from the sensors
  Broadcast a QueryOnPath message
  Listen for the first sensor to reply
  if a sensor replies with an OnPathAck message then
    Send a QueryPath message to that sensor
    // The sensor should reply with a list of PathSegments it is on
    if that sensor replies with a QueryAck message then
      Store the PathSegments from the QueryAck message in order of precedence.
  // Guide the robot
  if Robot has reached current Waypoint then
    Get next Waypoint from list in order of precedence
    Head for next Waypoint
```

After the initialization phase that places the robot on the path, the navigation guidance algorithm summarized as Algorithm 2 is used to control the motion direction of the robot. The robot starts by broadcasting a `QueryOnPath` message which includes the sender's id and location. A sensor on the path that receives this message replies with a `QueryAck` message which includes the path section, some consecutive way points, and an indication of where these way points fit into the path. By gathering lists of segments from multiple sensors the entire path can be assembled incrementally as the robot moves. Paths that cross themselves allow for some fault tolerance in the robot's knowledge of the path, since if the robot loses the path, it may have a future segment of it already stored if it has passed an intersection.

Once the robot has acquired path segments from a sensor, it can then arrange them in order of precedence and follow them in order. Thus the path itself is independent of the sensor's own location and can be specified to any level of precision needed.

4 Experiments

4.1 Experimental Testbed

The experiments were carried out on September 17, 2003 in the Planetary Robotics Building at CMU. We implemented the robot-assisted localization algorithm and the sensor-assisted guidance algorithm on an experimental testbed consisting of a sensor network with 54 Mica Motes [4, 5] and a flying robot. The flying robot consists of 4 computer controlled winches (implemented using Animatics Smart motors) located at the corners of a square with cables going up to pulleys at roof height then down to a common point above the 'flying' platform. The crane is controlled by a server program running on a PC. Commands and status are communicated using the IPC protocol(see www.cs.cmu.edu/afs/cs/project/TCA/www/ipc). The platform comprises a single-board Pentium-based computer running Linux, with an 802.11

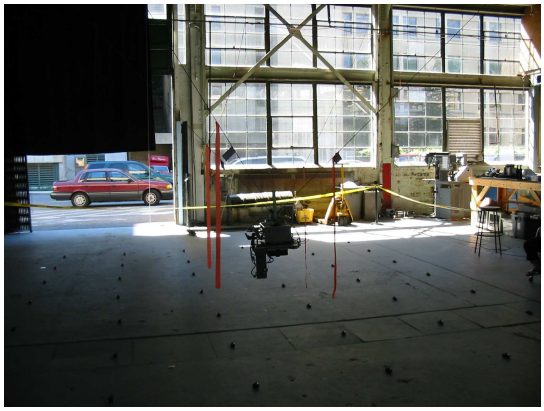


Fig. 2. The experimental testbed consisting of 49 Motes on the ground and the flying robot.

link and an on-board serially connected basestation Mote, to communicate with the sensor field. The robot has a workspace almost 10 m square and 4 m high.

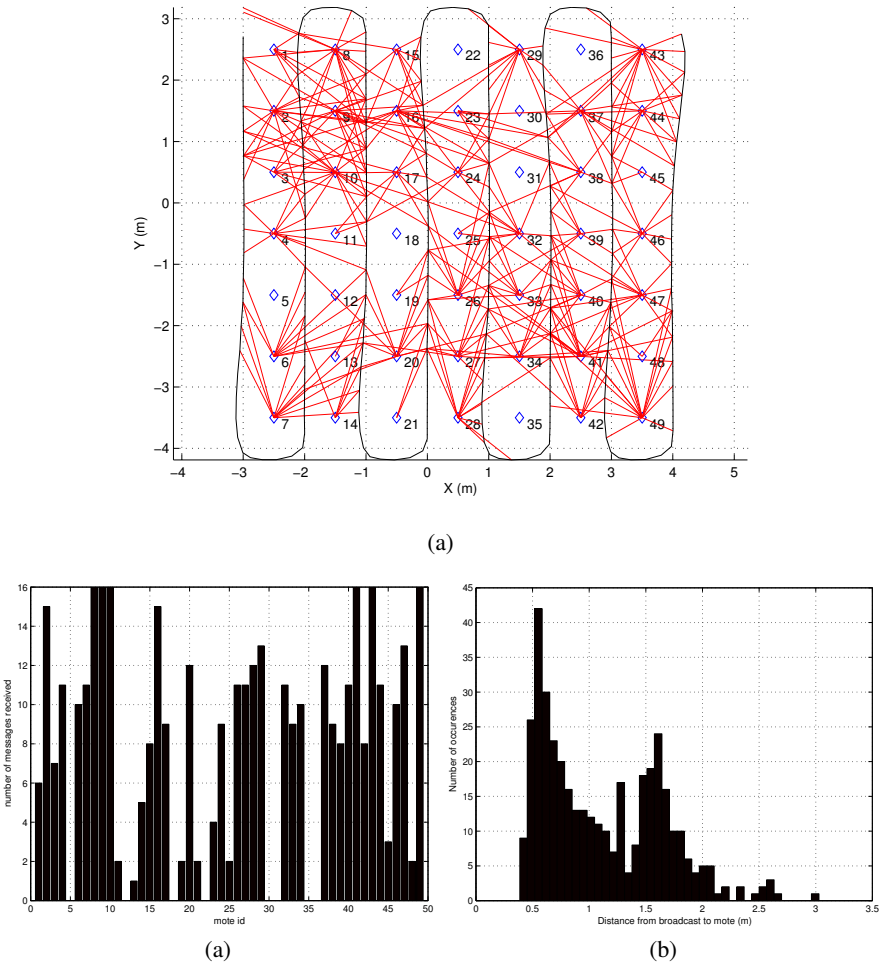
We used a 7x7 grid of sensors, laid out with a 1 meter spacing. The center of the grid was (0,0) and the sensors were placed starting 0.5 meters from the center, see Figure 3(a) where the diamonds represent the surveyed positions of the motes.

The Flashlight sensor interface [7] was used to adjust the RF power of the sensors in the grid to an optimal level for communication with the robot as it traveled 1-2 meters above the sensors (this was a trial-and-error adjustment, gradually incrementing the mote power until the robot was getting good communications) The motes ran TinyOS 0.6 with long (120 byte payload) messages.

4.2 Localization results

During localization the flying robot followed a preprogrammed serpentine path, see Figure 3(a). Once per second the flying computer obtained its current coordinate from the control computer using IPC over the 802.11 link, and broadcast this via the basestation mote. Each ground mote used the broadcasts to compute a centroid based location for itself. Figure 3(a) shows the robot path and the location of each of the broadcasts the motes received. It is clear that the motes do not receive messages uniformly from all directions, motes 6 and 7 are good examples of this. We speculate that this is due to the non-spherical antenna patterns for transmitter and receiver motes, as well as masking by the body of the flying platform itself. The motes received between about 2 and 16 broadcasts each as can be seen in Figure 3(b) with a median value of 10. Figure3(c) shows a histogram of the distances over which the broadcast messages were received, a maximum of 3m and a median of 1m.

Each mote computes its location using the centroid of all received broadcasts, but can store up to 200 localization broadcasts for download and analysis. Figure 4(a) shows the true and estimated mote locations. We can see a general bias inward

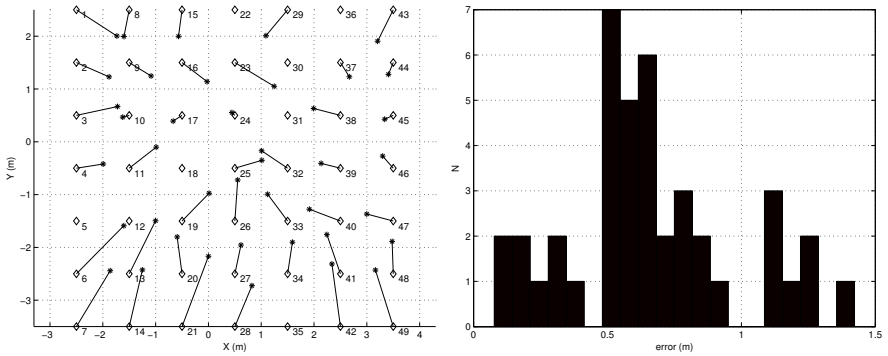


(a)

(a) (b)

Fig. 3. Localization results. (a) Mote field showing path of robot and broadcast positions, and all broadcasts received. (b) Number of localization messages received by each node. (c) Histogram of distances from mote to broadcast.

and this would be expected given the the bias in the direction from which broadcasts were received. Figure 4(b) shows a histogram of the error magnitudes and indicates a maximum value of 1.4m and a median of 0.6m which is approximately half the grid spacing. This level of performance matches our previous results obtained with experiments with a real helicopter and differential GPS [2].



(a) (b)

Fig. 4. Localization performance using centroid method. (a) Actual (◇) and estimated (*) location. (b) Histogram of error vector length.

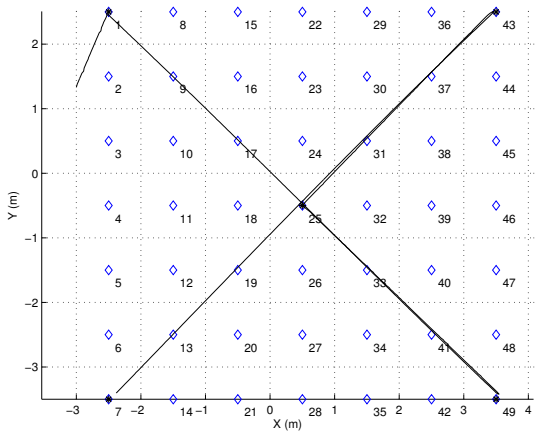


Fig. 5. Path following performance. The actual path followed by the robot is shown in black, and the asterisks indicate waypoints. The path started at node 7.

4.3 Path following results

Once localized, a PATH message was sent from a basestation to establish a path through the mote field. The PATH message propagated using the algorithm described in [2]. Then the robot was turned loose in a path following mode, using the path following algorithm in [2]. It queried for path waypoints and built up a list of waypoints as it followed the path. We experimented with a square path (around the border of the grid) and an X shaped path (corner to center to corner). The robot followed both types of path perfectly. Even though the localization of the motes was not perfect, it was sufficient to support the geographic routing of the PATH message with a 1 meter width. The actual path itself was stored as perfectly precise information in these

motes and hence the robot was able to get precise waypoints to follow, resulting in perfect path following (within the tolerances of the system) as shown in Figure 5.

Since there were multiple motes along each segment of the path, there was redundant information in the sensor field in case any of the motes were not working (and as it later turned out about 6-7 of them were not during each test, either due to defunct radios, or due to not hearing any messages for other reasons.)

5 Conclusion

We have described how robots and sensor networks can function synergistically to perform tasks such as localization and guidance. Simulation studies provide insight into the achievable performance of various localization methods, and experimental results are provided. The localization approach does not require inter-sensor communications. New algorithms for path following are presented along with experimental validation.

Acknowledgments

This work is done as part of the First Responder Project at the Institute of Security Technology Studies at Dartmouth College and is a collaborative project between the labs at Dartmouth, MIT, and CSIRO. The authors would like to thank Dr. Sanjiv Singh for facilitating the physical experiments at CMU. Support for this work has also been provided in part through the NSF award 0225446, ONR award N00014-01-1-0675, the Darpa TASK program, MIT project Oxygen, and Intel.

References

1. N. Bulusu, J. Heidemann, and D. Estrin. Adaptive beacom placement. In *Proceedings of the* 21^{st} *Conference on Distributed Computing Systems*, Phoenix, AZ, 2001.
2. P. Corke, R. Peterson, and D. Rus. Networked robots: Flying robot navigation with a sensor net. In *Proc. of the 2003 International Symposium on Robotics Research*, Siena, Italy, 2003.
3. A. Galstyan, B. Krishnamachari, and K. Lerman. Distributed online localization in sensor networks using a moving target. submitted to 2003 acm senssys.
4. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, 2000.
5. Jason Hill, Philip Bounadonna, and David Culler. Active message communication for tiny network sensors. In *INFOCOM*, 2001.
6. Qun Li, Michael de Rosa, and Daniela Rus. Distributed algorithms for guiding navigation across a sensor net. In *Proceedings of MobiCom 2003*, 2003.
7. Ron Peterson and Daniela Rus. Interacting with a sensor network. In *Proceedings of the 2002 Australian Conference on Robotics and Automation*, Auckland, NZ, November 2002.
8. S. Simic and S. Sastri. Distributed localization in wireless sensor networks, Available at citeseer.nj.nec.com/464015.html, 2002.