Peter Corke

CSIRO ICT Centre, Australia

Ron Peterson

Dartmouth Computer Science Department and
The Institute for Security Technology Studies (ISTS) at
Dartmouth College
Hanover, NH 03755, USA

Daniela Rus

Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology (MIT)
Cambridge, MA 02139, USA
rus@csail.mit.edu

Localization and Navigation Assisted by Networked Cooperating Sensors and Robots

Abstract

In this paper we discuss how a network of sensors and robots can cooperate to solve important robotics problems such as localization and navigation. We use a robot to localize sensor nodes, and we then use these localized nodes to navigate robots and humans through the sensorized space. We explore these novel ideas with results from two large-scale sensor network and robot experiments involving 50 motes, two types of flying robot: an autonomous helicopter and a large indoor cable array robot, and a human–network interface. We present the distributed algorithms for localization, geographic routing, path definition and incremental navigation. We also describe how a human can be guided using a simple hand-held device that interfaces to this same environmental infrastructure.

KEY WORDS—distributed robots, networked robots, localization, navigation, sensor nets

1. Introduction

Recent advances in sensor networks point towards a future with thousands of small low-cost sensors embedded in the environment. Devices such as the Mica Mote (Hill et al. 2000; Hill, Bounadonna, and Culler 2001), the single chip "Spec" (http://robotics.eecs.berkeley.edu/~pister/smartdust/), and AutoId are steps along the path to the ultimate goal of smart dust. We believe that these technologies will have a profound effect on robotics as we know it today, and importantly that robotics will have a profound effect on networks of sensors. Today's robots are large complex systems with a small number of expensive sensors of limited spatial reach. The new paradigm is of ubiquitous sensors embedded in the environment with which the robot interacts: to deploy them, to harvest data from them, and to task them. In turn, the embedded sensors can provide the robot with models that are highly adaptive to changes in the environment and can task and retask the robots using feedback from the sensors.

In this paper we explore the possibilities with this new way of robot and sensor interaction. We are particularly concerned with how a network of sensors and robots can cooperate to solve important robotics problems such as localization and navigation. The research challenges are at the intersection between communication, control, and sensing. Networked communication can be used to enhance the perceptual field of the robots and thus can have a profound impact on the robot's ability to adapt fast to changes in its environment and task, even when the changes are outside the robot's perception range. Networked communication also plays a role in computing adaptive and up-to-date environmental maps, supporting real-time recording of events outside a robot's perceptual field. In order to enable this type of application, the robots and sensors have to be connected as a network and each node in the system has to be aware of its location. The mobility and overall computation power of robots enable the system of robots and sensors to compute and maintain the location of each node in the system. Controlled mobility also allows the system to ensure that the network is fully connected and the sensors cover the desired space.

Specifically, we explore the synergies between communication, perception, and control, and show how mobility can assist and enhance the localization capabilities of sensor networks and how networked communication can assist and enhance the navigation capabilities of autonomous robots. We

bring together ideas from research fields such as sensor networks, mobile computing and robotics, and present some results that are not achievable by robot or sensor alone. We use a robot to localize sensor nodes, and we then use these localized nodes to navigate robots and humans through the sensorized space.

We consider a robot network to be a collection of robots distributed over some area that form an ad hoc network—a network formed without the aid of any established infrastructure or centralized administration. Such a network can support robot–robot and sensor–robot communications, or even human–human, human–robotic, and sensor–human communications for scenarios in which the network provides the communication backbone, e.g., for search and rescue and first responders. Such systems, by virtue of having no central control, are robust and well suited for tasks in extreme environments. The nodes of our network will likely be heterogeneous and include mobile robots, mobile and static sensors, even people or animals. Each sensor has communication capability, limited memory and processing capabilities, and multiple sensing modalities. Thus, we extend the notion of sensor networks which has been studied by the networking community for static sensors to networks of robots that have natural mobility.

Navigation is an example of how simple nodes distributed over a large geographical area can assist with global tasks. The nodes sample the state of the local environment and communicate that to nearby neighbors, either continuously or in the event of some significant change. Hop-by-hop communication is used to propagate this information and distribute it throughout the network. For example, consider dispersing a sensor network over a large forest to monitor forest fires. The sensors could be dropped from a flying robot and localize using GPS locations broadcast by the robot. Once localized, they could sense and propagate temperature levels to compute a temperature gradient for the region. The occurrence of a new fire would be signaled automatically across the network. In addition, the sensor network can locally compute the shortest path to the fire to guide firefighters, and indicate the safest path to exit for other people. As the fire progresses, the safest path shifts and changes and the network is capable of computing it in real time. The sensor network can update these paths in real time, accommodating changes due to environmental conditions such as shifting winds. The same information can be used to guide search and rescue teams to the humans along different paths. Thus, multiple goals and paths can coexist within the system. The capabilities of robots and people are extended through interaction with the network, extending their senses and ability to act over a massive area.

The robot may also inject data into the network based on its superior sensory or reasoning capability, for example configuring the network by reprogramming its nodes, synchronizing clocks, deploying new sensors to fill in communication gaps, or calibrating sensors by transmitting reference values sensed by the robot itself. The ability to retask and reposition sensors in a network by sending state changes or uploading new code greatly enhances the utility of such a network. It allows different parts of the network to be tailored to specific tasks, capabilities to be added or changed, and information to be stored in the nodes in the network.

To realize the vision of cooperating robots and sensor networks for such applications, several aspects of the problem must be addressed. (1) The robot and sensor teams have to first be deployed. (2) Once at the destination, the nodes must be localized so that the data they collect can be associated with specific geographic locations. (3) The localized nodes should have the ability to monitor and detect events. (4) The localized nodes should cooperate to stitch their local data into global maps. (5) When detecting an event, the system should compute a path from the current location of the nearest flying robot to the event location. This path should be adaptive and shift according to detectable events in the system. (7) Given a path embedded in the network, the network should be able to guide the motion of the flying robot incrementally. The incremental nature of navigation allows the robot to adapt to events outside its perception range. (8) The robot and sensor network should adjust their locations in a way that ensures the space is covered and the network is fully connected.

In this paper we present two examples of cooperation between a sensor network, robots and people: robot-assisted localization and communication-assisted navigation. These concepts have been experimentally validated with a physical sensor network consisting of 54 Mote sensors (Hill et al. 2000; Hill, Bounadonna, and Culler 2001) and two types of flying robot.

The paper is organized as follows. In the remainder of this section we outline related work and describe the experimental setup we use. Then we present examples of some useful tasks that can be performed by robot and sensor network cooperation—more are possible. In the following sections we discuss, in turn, robot-assisted localization, communications-assisted path routing, and network-assisted navigation. For each of these topics we present algorithms, simulation and experimental results, and discussion.

1.1. Related Work

Sensor networks are ad hoc networks, built without any existing infrastructure, where each node can sense, compute, and communicate to nearby neighbors. Mobile robot networks are sensor networks whose nodes move under their own control. Massively distributed sensor networks are becoming a reality (Hill et al. 2000). (See http://www.cast.cse.ohio-state.edu/exscal/index.php?page=main for a description of a 1000 node deployment.) Important contributions on which this work builds include Pottie (1998), Agre and Clare (2000), Chang and Tassiulas (1999, 2000a, 2000b), Singh, Woo, and Raghavendra (1998), Estrin et al. (1999), Li, de Rosa, and Rus (2003), and Das et al. (2003). Sensor network mobil-

ity issues are discussed in Batalin and Sukhatme (2005). Other key results in controlling sensor networks include node design, routing, control of information gathering, representation of information, and in-network information processing (Hill et al. 2000; Ramanathan and Hain 2000; Chen et al. 2001; Wattenhofer et al. 2001; Xu, Heidemann, and Estrin 2001; Chen and Hudson 2002; Guibas 2002; Li et al. 2002; Pradhan, Kusuma, and Ramchandran 2002; Zhao, Shin, and Reich 2002). Much work in sensor networks builds on results in ad hoc networks that address the limitations of wireless networks, i.e., low bandwidth, high error rates, low power, disconnections (Karlin and Taylor 1975; Cheng et al. 1989; Johnson and Maltz 1996; Murthy and Garcia-Luna-Aceves 1996; Basch, Guibas, and Zhang 1997; Haas 1997; Kotz et al. 1997; Basagni, Chlamtac, and Syrotiuk 1998; Gupta and Kumar 1998; Ko and Vaidya 1998; Singh, Woo, and Raghavendra 1998; Okino and Cybenko 1999; Li, Aslam, and Rus 2001).

The node localization problem has been previously discussed by others and usually requires estimates of inter-node distance, a difficult problem. Simić and Sastry (2002) present a distributed algorithm that localizes a field of nodes in the case where a fraction of nodes are already localized. Bulusu, Heidemann, and Estrin (2001) propose a localization method that uses fixed beacons with known position. Galystyan, Krishnamachari, and Lerman (2003) have 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, pre-localized nodes, or inter-node communications. In reality, the communications region has a complex non-circular shape and the probability of message reception, as well as signal strength varies in a complex manner with distance (Rappaport and Sandhu 1994). These observations accord with our experimental experience. The results reported to date have been based on simulation and assume a circular radio communications region which is far from reality (Kotz et al. 2004).

1.2. Experimental Setup

Our algorithms for robot-assisted localization and communication-assisted navigation have been implemented using a sensor network of Mica Motes and two different flying platforms: an autonomous helicopter and a cable array robot. We also implemented the communication-assisted navigation algorithms for humans who can interact with a sensor network using a hand-held device called a Flashlight. The hardware is described next.

1.2.1. Sensor Network Hardware

Our algorithms are hardware-independent but the message formats used by the networked system are hardware-dependent. We use a sensor network that consists of 54 Mica Motes (Hill et al. 2000; Hill, Bounadonna, and Culler 2001); see Figures 1 and 2. Each node contains a main processor and

sensor board. The Mote handles data processing tasks, A/D conversion of sensor output, RF transmission and reception, and user interface I/O. It consists of an Atmel ATMega128 microcontroller (with 4 MHz 8-bit CPU, 128KB flash program space, 4K RAM, 4K EEPROM), a 916 MHz RF transceiver (50 Kbits/s, nominal 30 m range), a UART and a 4 Mbit serial flash. A Mote runs for approximately one month on two AA batteries. It includes light, sound, and temperature sensors, but other types of sensors may be added. Each Mote runs the TinyOS 0.6 operating system with long (120 byte payload) messages. The sensors are currently programmed to react to sudden increases in light and temperature but other sensory modes are possible.

1.2.2. Flying Robot

In this work we used two flying platforms. The first is the CSIRO helicopter (see Figure 1), which is a hobby type (60 class) JR Ergo, and has a limited, 5 kg, payload capability (Buskey et al. 2003). This helicopter differs from other similar projects in using low-cost sensors for control. These include a custom inertial measurement unit, magnetometer and a vision system. The control computer is an 800 MHz P3 with solid-state disks running the LynxOS operating system. On-board application software interacts with the sensor network on the ground by means of a serial connection to a base-station Mote fitted to the underside of the helicopter.

The second flying robot is a cable array robot, which we refer to as the flying robot simulator. It comprises four 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 (Simmons and James 2001). The platform comprises a single-board Pentium-based computer running Linux, with an 802.11 link and an on-board serially connected base-station Mote, to communicate with the sensor field. The robot, located within the Planetary Robotics Building at Carnegie-Mellon University (CMU), has a workspace almost 10 m square and 4 m high.

1.2.3. Human Interface

The sensory Flashlight (see Figure 2) is a hand-held device which uses the metaphor of a flashlight to provide a human interface to the network by means of visible and vibratory cues (Peterson and Rus 2002). The Flashlight consists of an analog electronic compass, alert LED, pager vibrator, a three position mode switch, a power switch, a range potentiometer, some power conditioning circuitry, and a microcontroller based CPU/RF transceiver. The processing and RF communication components of the Flashlight and the sensor network are Berkeley Motes. With suitable software, the Flashlight

Fig. 1. Left: helicopter in the air over the outdoor sensor network consisting of 54 Motes (Hill et al. 2000; Hill, Bounadonna, and Culler 2001). The Motes sit on top of the dark flower pots. Right: the experimental testbed consisting of 49 Motes on the ground and the flying robot simulator.
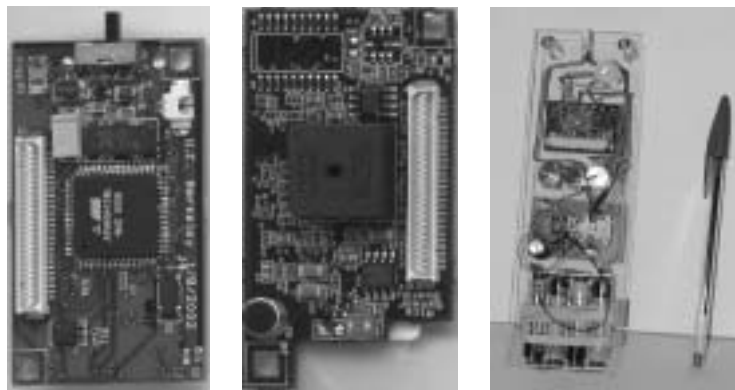

Fig. 2. From left to right: a Mote board, Mote sensor board, and the Flashlight device.

can perform a variety of functions to mediate between the user and the sensor network, sending commands to the sensor network and receiving/displaying information from the network. The potentiometer and three position switch can be programmed to control any variable parameter such as setting software mode of operation, setting radio transmit power, defining a maximum range for messages transmitted from the Flashlight, or setting an alert sensitivity threshold. The silent vibrating alarm and LED can be activated to signal alerts received from the sensor network. The LED intensity and vibrator vibration amplitude can be pulse modulated to encode additional information about the sensor space. The compass can be used to limit alerts received to those originating from the direction the Flashlight is currently pointing, or to send geographically routed messages out along directions towards which the Flashlight is pointed. Typical uses for the Flashlight include: activating sensors towards which it is pointed, but only beyond the range set on the potentiometer; experiment start/stop/reset control; reconfiguration of the transmit power
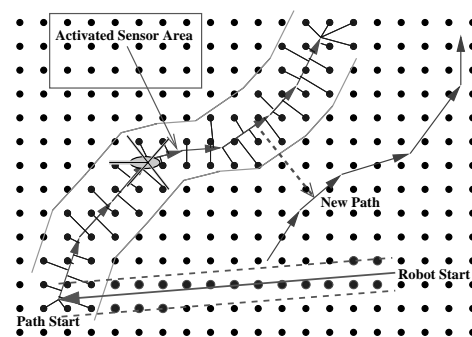

Fig. 3. A sensor network with a path marked by sensor nodes. In response to an environment trigger, the sensor network computes a new path for the helicopter and an intermediate path to guide the helicopter to the new path.

of a sensor network; alerting the user when there is a sensor indicated alarm in the direction the flashlight is pointed.
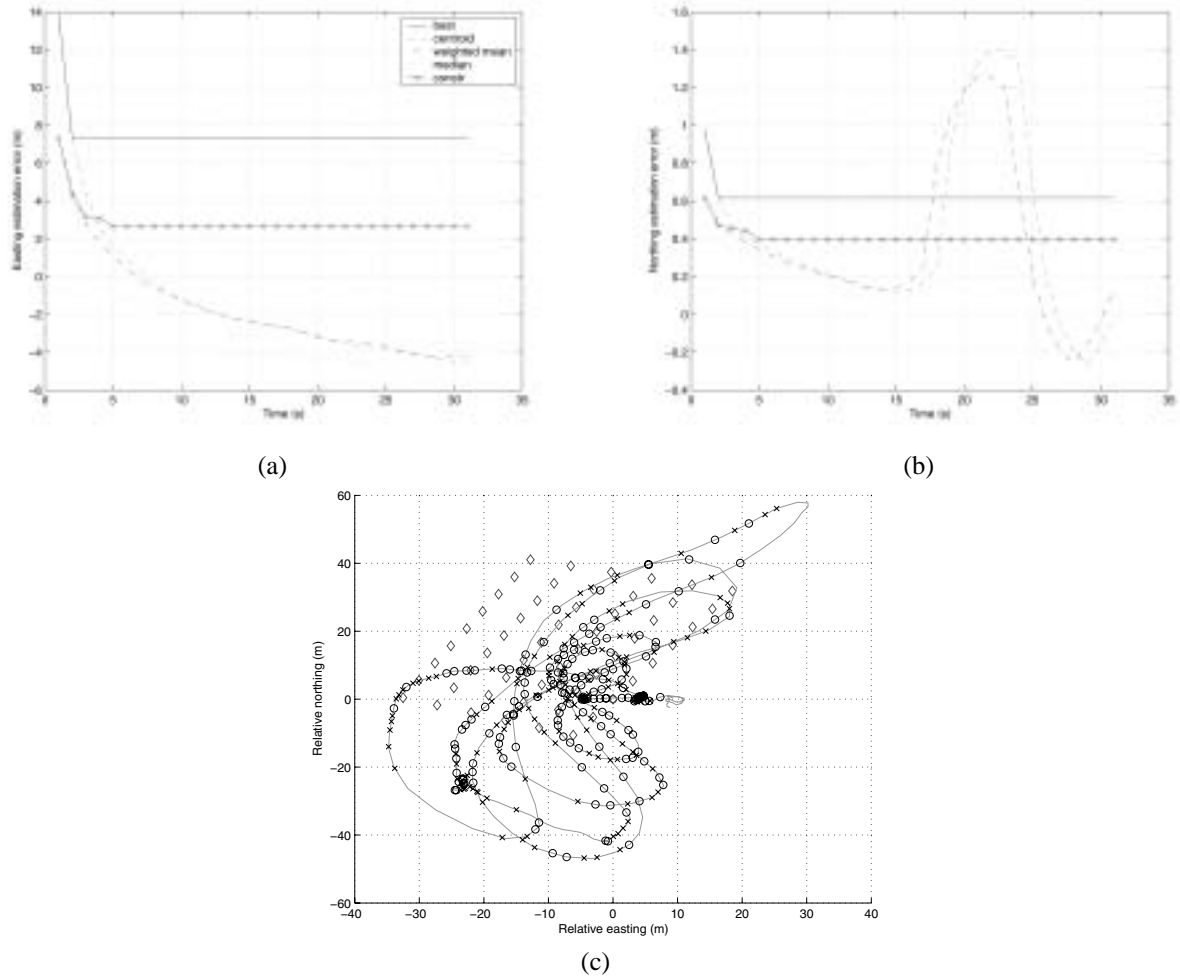
(a)

(b)

(c)

Fig. 4. Results of off-line localization by GPS, evolution of different estimates with time for our five localization methods. Error in the Easting (a) and Northing (b) directions are shown. (c) The helicopter path is shown with GPS reception marked: **o** denotes a good packet and **x** a bad packet.

1.2.4. Experimental Sites

In March 2003 we conducted outdoor experiments with the robot helicopter and 54 Mica Motes (see Figure 1) at the CSIRO laboratory in Brisbane. The Motes were placed at the nodes of a 6 m grid on a gentle slope. The grid was established using tape measures and the corner points were surveyed using differential GPS, and the coordinates of the other points were interpolated.

Experiments showed that the radio range of the Motes was very poor outdoors and this is discussed further in Section 5. A base-station Mote connected to a laptop was used to control the Mote network. Figure 4 shows the layout of the Motes, represented by diamonds overlaid with the flight path of the robot.

In September 2003 we conducted a second round of experiments at CMU using the flying robot simulator. A simple localization algorithm and the sensor-assisted guidance algorithm was tested with a sensor network comprising 49 Mica Motes. We used a 7×7 grid of sensors, laid out with a 1 m spacing (see Figure 5(a)), where the diamonds represent the surveyed positions of the Motes.

2. Robot-assisted Localization

In Corke, Peterson, and Rus (2003) we introduced the idea of robot-assisted localization, an approach to localization that is orthogonal to the previous work in localization in that it does not require inter-node communication and is suitable for sensor networks deployed outdoors. Once we have the ability to localize deployed sensors we are able to employ efficient routing techniques such as geographic routing, as well as using the network to guide the robot. Localization also
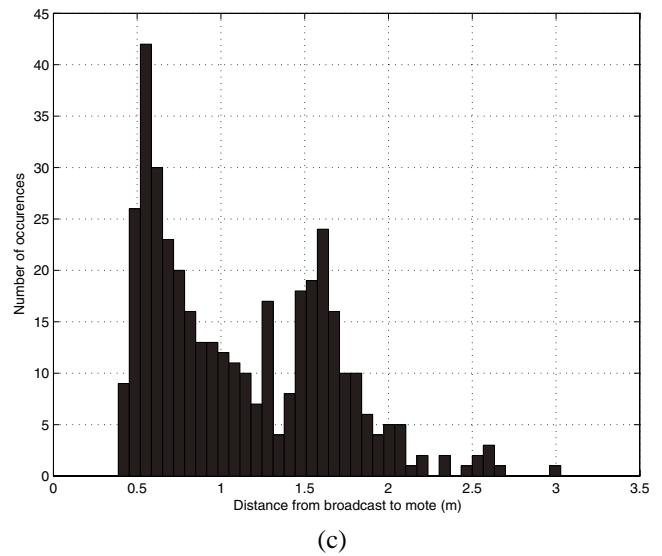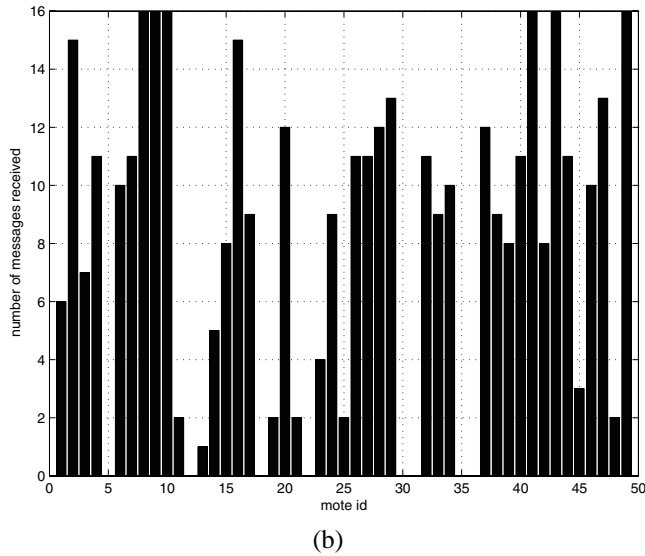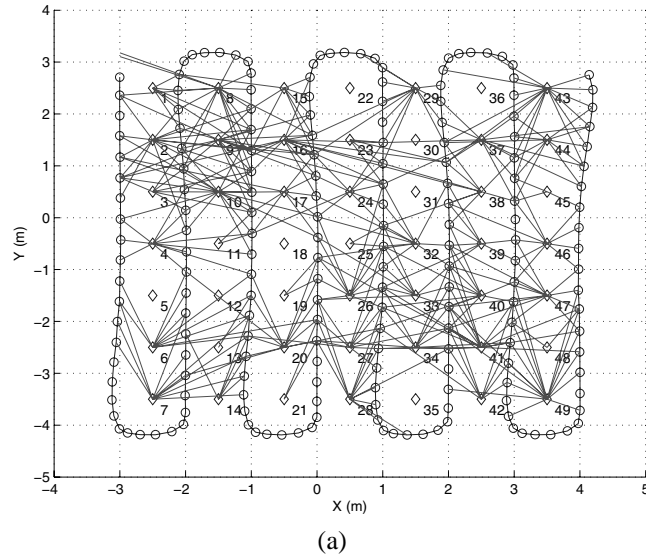
(a)

(b)

(c)

Fig. 5. 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.

increases the value and usefulness of sensed data by tagging events to geographic location.

2.1. Approach

We assume that the sensors have been deployed from the robot in a way that covers the area of interest uniformly but not necessarily regularly. For very large sensor networks, the localization requirement could be limiting since it is 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 fly-

ing 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 incrementally process all broadcasts they receive to refine their estimated location. The mobile node's broadcast messages contain its position $p_i = (x_i, y_i)$ and sensors receive the message with signal strength s_i or not at all. Each sensor listens for the broadcasts and improves its location estimate over time using one of the following six algorithms. We will assume, simplistically (Kotz et al. 2004), that the maximum radio range projected onto the ground plane is R.

strongest. Assume that the strongest received message so far is the best estimate of node position, since it was sent when the robot was nearest.

> **if** $s_i > s_{max}$ **then**
>
>> $s_{max} = s_i$
>>
>> $\hat{p} = p_i$
>
> where s_{max} is initialized to zero.

mean. Assuming that the receiver reception pattern is a disk and that the robot position is uniformly distributed within that disk, we can estimate the sensor position by the mean robot position $\hat{p}_i = \Sigma_i p_i / N$

wmean. A refinement of above and increasing the significance of positions broadcast from nearby, we use the signal strength weighted mean of the received position as the estimate $\hat{p}_i = \Sigma_i s_i p_i / \Sigma_i s_i$

median. The median statistic has robustness to outlier data $\hat{p}_i = \text{median}(p_{1 \cdots i})$

constraint. Consider each received position as a constraint (Galystyan, Krishnamachari, and Lerman 2003) on the node position which is considered to lie within the rectangular region Q. At each step we constrain the node to lie in the intersection of its current region, $Q(k)$, and a square region of side length $2d$ centered on the GPS transmission, that is, $Q(k+1) = Q(k) \cap [x(k) - d, x(k) + d] \times [y(k) - d, y(k) + d]$. The position estimate of the node is taken as the centroid of the region $Q(k)$. The parameter d should reflect the size of the radio communications region, i.e., $d = R/2$. $Q(0)$ is initialized to a square of side length R.

bound. Consider n directions defined by unit vectors $\mathbf{u}_i \in \Re^2$. For each broadcast, $p_i \in \Re^2$ we update $m_i(k+1) = \max m_i(k), p_i \cdot \mathbf{u}_i$ the maximum distance along direction $\mathbf{u}_i$ that a message was received. The position estimate of the node is taken as the mean $\hat{p} = \Sigma m_i \mathbf{u}_i / n$. The simplest case is for four $\mathbf{u}_i$ each 90° apart.

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 range, R. Algorithms **constraint** and **bound** are similar in estimating a bound on the node's location: **constraint** estimates a minimum bound, whereas **bound** estimates the maximum bound on radio reception. The **constraint** method has a parameter which needs to be adjusted. Algorithm **median** has the disadvantage of needing to store all messages which may be problematic on memory limited hardware.

The maximum error bound for all these methods is R and will occur when a message is received from the robot at maximum radio range. If R is small, then the maximum error will be small, but the likelihood of the robot being within radio range is low. Increasing R to ensure reception of messages from the robot will increase the error bound.

We can characterize the error in the localization analytically for a simplified case when the path of the robot is a grid shape and the communication for each sensor is modeled as a unit disk as follows.

THEOREM 1.   Suppose the flying robot travels an exact grid of with spacing g and has the ability to broadcast its exact coordinates at each grid point. Suppose further that the communication range is a disk of radius R. The localization error of the averaging algorithm is at most $(1/4)g$ in each of the x and y directions and $(\sqrt{2}/4)g$ overall.

Proof. The goal is to show that the location broadcasts generated in a grid pattern give an error whose size can be characterized. Imagine the grid formed by the flying robot's location broadcasts. Consider a sensor node in the field. The location of its communication disk can be anywhere within the grid field.

Suppose the grid size is some fraction of the radius of the disk, so that several localization broadcasts are inside the disk. If the sensor disk is perfectly aligned (that is, one location ping hits the center of the disk and the other location broadcasts line up nicely and evenly along the two axes), the localization error is zero because in the computation of the average all the offset values cancel out. Now suppose the location of the disk is shifted so that the center location broadcast is no longer at the disk center. Suppose that this offset is such that no localization points exit the disk and no new ones come into the disk. How far can we shift? Consider another limit case where the center of the disk is in the middle/center of four localization points, again a case where there is zero localization error using the average computation. This case happens when the localization pings are off by $(1/2)g$. So it follows that the maximum error is for displacements between 0 and 1/2, which is maximized at $(1/4)g$. This bound characterizes the error in each of the x and y axes. Thus, the overall error is bounded by $\sqrt{x_{error}^2 + y_{error}^2}$. □

2.2. Experimental Results

In this section we compare empirically the performance of the five[1] different approaches to localization introduced in Section 2 using data acquired during our first experiment with a flying robot. We consider only the case of the node at the origin since the results of others are similar. The error between estimated and actual Mote coordinate for each of the algorithms is shown in Figure 4. The results have been computed off-line using GPS coordinates obtained by each Mote from the actual helicopter path shown in Figure 4. The parameters used were $d = 5$ m and $s_{min} = 470$. We can see that the

---

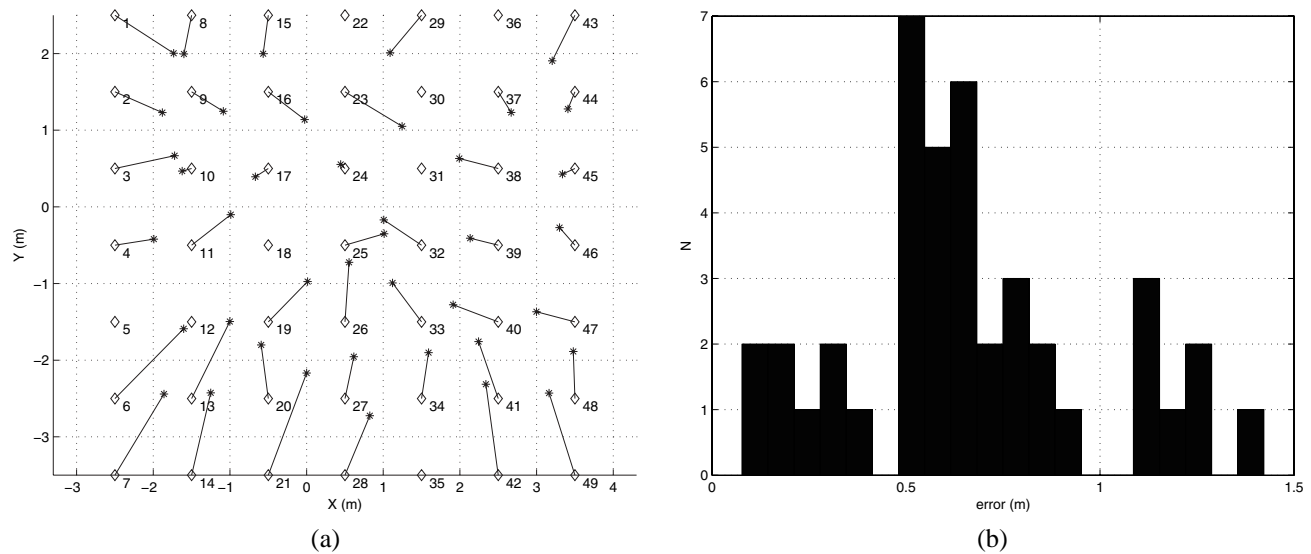1. The **bound** algorithm was developed subsequently.

Fig. 6. Localization performance using centroid method. (a) Actual (◇) and estimated (*) location. (b) Histogram of error vector length.

mean and weighted mean are biased, particular in the East-ing direction, due to the path taken by the helicopter and/or non-symmetry in the Mote's radio reception footprint. The method **strongest** is simple but has high residual error. The **median** does not perform significantly better than the **mean** or **wmean** estimates. In the Northing direction we can see that the mean and weighted mean methods were strongly biased around $t = 20$ s due to the robot staying on one side of the radio's reception field. The **constraint** method was arguably the best performer and is computationally cheap, although it is highly sensitive to the choice of d which should reflect actual radio range.

The errors shown should be considered with respect to the accuracy of differential GPS itself which is of the order of sev-eral meters. Achievable localization accuracy is of the order of one-half of the grid spacing, which is more than sufficient to enable the geographic routing strategies discussed herein. We note that the methods do not require a range estimate derived from signal strength, a difficult inverse problem (Rappaport and Sandhu 1994), and do not make any assumption about the size or shape of the radio communications region.

In the second experiment, with the flying robot simula-tor at CMU, the robot followed a serpentine path (see Fig-ure 5(a)). Once per second the flying computer obtained its current coordinate from the control computer using IPC over the 802.11 link, a virtual GPS, and broadcast this via the on-board base-station Mote. Each ground Mote recorded all the X, Y broadcasts it received and used the **mean** method to es-timate its location. Figure 5(a) shows the robot path and the locations from which the position broadcasts were made. It is clear that the Motes do not receive messages uniformly from all directions; Motes 6 and 7 are clear examples of this. We

speculate that this is due to the non-spherical antenna patterns for transmitter and receiver Motes, as well as masking of some ground Motes by the body of the flying platform itself. Eight Motes received no broadcasts at all due to networking errors, packet loss, or Mote hardware failures. The remaining Motes received between two and 16 broadcasts each, as can be seen in Figure 5(b) with a median value of 10. Figure 5(c) shows a histogram of the distances over which the broadcast messages were received, a maximum of 3 m and a median of 1 m.

Each Mote computes its location using the centroid of all received broadcasts, but can store up to 200 localization broad-casts for later download and analysis. Figure 6(a) compares the true and estimated (using mean method only) Mote lo-cations. We can see a general bias inward and this would be expected given the bias in the direction from which broad-casts were received. Figure 6(b) shows a histogram of the error magnitudes and indicates a maximum value of 1.4 m and a median of 0.6 m, which is, again, approximately half the grid spacing that we achieved with the real helicopter and differential GPS (Corke, Peterson, and Rus 2003).

2.3. Discussion

In the robot-assisted localization algorithm, the robot regu-larly broadcasts its location. When within the reception range of the sensor, these broadcasts provide input to the localiza-tion algorithm. The reception range is not symmetrical due to the directional sensitivity variation of both the transmitting and receiving radios involved, terrain, etc. Since the asym-metry depends on the relative orientation of both antennas, it will vary from encounter to encounter, which highlights the following two problems.

1. The asymmetry is not known a priori, 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 5(a) shows an extreme case of directional reception in which this assumption fails.

2. With relatively few measurements occurring within the reception range the estimate of centroid is likely to be biased.

The first problem is not solvable given current radios; multiple encounters at different relative antenna orientations might provide some remedy, but would increase the time and cost of any post-deployment localization phase. Some possible ways to ameliorate the second problem include the following.

1. Increasing the rate at which position broadcasts are sent, giving more samples within the reception range, and improving the estimate of the centroid.

2. Increasing the reception range, R, 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. Decreasing the reception range, perhaps combined with improvement number 1, so that those broadcasts that are received are very close to the location of the sensor. Of course, this increases the possibility that a node will receive no broadcast at all.

To investigate the efficacy of such improvements we have conducted numerical experiments (Extension 2) in which we vary the rate at which the robot broadcasts its position, and the radio reception range. To eliminate the problem of path dependence while testing postulates (1)–(3) above, our simulation uses a fixed serpentine robot path and 100 sensors deployed randomly with a uniform distribution in a square region 100×100 m^2. The robot starts at the origin in the lower-left corner, moves 100 m to the right, up 20 m, 100 m to the left, then up another 20 m, and repeats the cycle. The mean inter-node spacing is 17 m. The radio propagation model assumes that signal strength decreases with distance and becomes zero at the maximum distance parameter which we can also vary.

For each numerical experiment we randomly deploy the sensors, then for each node, we run the six localization algorithms with a particular set of simulation parameters, such as radio range and broadcast rate. For the **constraint** method we set $d = 20$. The mean and maximum localization error statistics for all the nodes are then computed. We repeat the experiment 100 times, and compute second-order statistics:

mean and standard deviation of the single experiment mean, as well as the maximum of the single experiment maximums.

Figure 7 shows some of the results. We observe that as the number of broadcasts increases (i.e., broadcasts are closer together) the localization error decreases and reaches a plateau at around 5 m or better. The method **strongest** performs least well, and the methods **constraint** and **bound** perform identically since the actual and assumed transmit radii are equal. Good performance is obtained for most methods with 100 broadcasts along the path. The total path length is 600 m, which means one broadcast every 6 m of travel, or once per second with a ground speed of 6 m s^{-1} or 21 km h^{-1}.

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 **constraint**, previously a strong performer, breaks down when the actual and assumed transmit radii are not equal. The best performer in this test is **wmean**, although **mean** and **bound** also behave well.

We plan to extend these numerical experiments to include stochastic packet reception models and non-symmetric radio reception models.

3. Communication-assisted Path Computation

Algorithm 1. The path routing algorithm.

```
NewPathFlag = FALSE
if a PathMessage is received then
    // Ignore the message if it has already been seen. I.e., we
    // are seeing the same message resent from another sensor.
    if PathMessage.MessageID ! = oldMessageID then
        oldMessageID = PathMessage.MessageID
        // Check if this sensor is on the path.
        while there are PathMessage.PathSegments left in the
        PathMessage
        do
            Calculate minimum Distance from PathMessage.
            PathSegment to this Sensor
            if Distance < PathMessage.PathWidth then
                // This sensor is on the Path
                First time here, erase previously stored path
                NewPathFlag = TRUE
                Rebroadcast the PathMessage
                Activate this sensor for robot guidance
                Store PathSegment
                SegmentCount++
    if NewPathFlag == FALSE then
        // This sensor is not on the path. Check if it should
        // forward the message towards the path.
        Compute heading1 from Sender to this sensor.
        Compute heading2 from Sender to start of path.
        Compute distance between this sensor and vector from Sender to
        start of path.
        if (abs(heading1 − heading2) < THRESHOLD) && (distance <
        SETWIDTH) then
            // This sensor is in the direction of the start of path.
            Rebroadcast the PathMessage.
```

In this section we consider storing path data in the sensor network to guide robots and humans. When possible, sending path data directly to a robot or human carried device is the
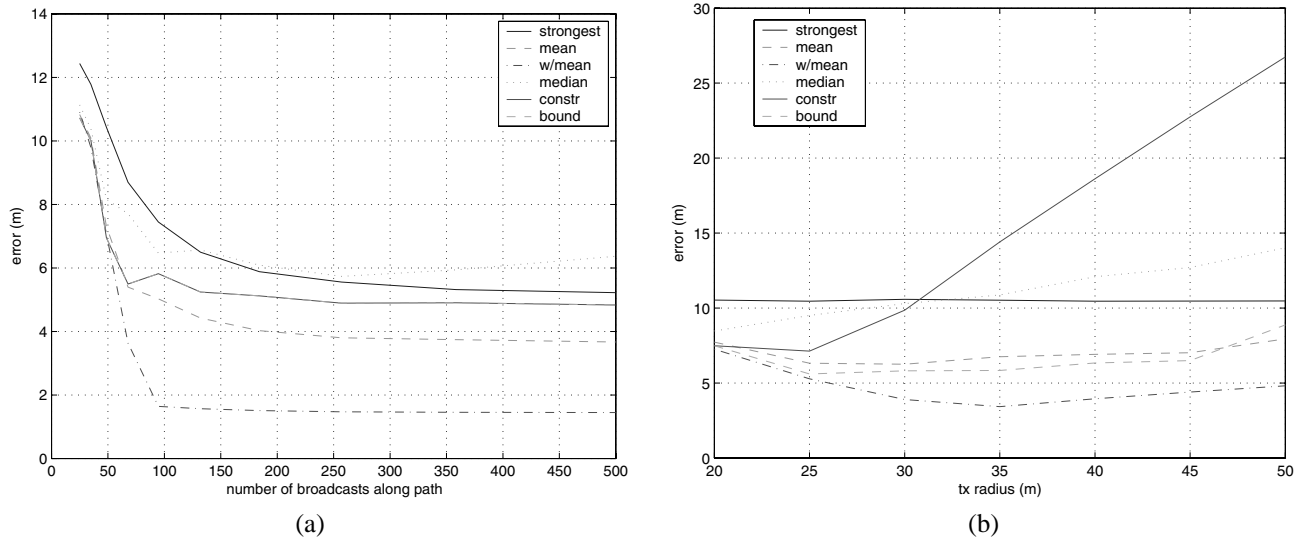
(a)

(b)

Fig. 7. Mean localization error for all the nodes from the Monte Carlo study using the six methods of Section 2. (a) Effect of varying the broadcast interval (transmit range = 20 m). (b) Effect of varying the transmission radius (50 broadcasts along path).

best way to provide guidance. However, there are many situations where path data need to reside in the sensor network, such as when the path computation is an incremental interaction between the robot and the sensor network or when it is desired that the location of the human clients not be revealed by routing a message to them over a long distance. There are also many ways to turn raw sensed data into guidance information represented as paths. We have chosen to focus on two methods. The first is a distributed path computation algorithm, which we previously developed (Li, de Rosa, and Rus 2003) and which can monitor the environment and encode a map of the environment in sensor space. Such a map can be constructed incrementally and adaptively as an artificial potential field using hop-by-hop communication. Areas of the sensor network where sensors have detected events can be represented as obstacles and have repulsing potential values while the goal has an attracting value. The potential field is computed by the obstacle and goal sensors diffusing information to their neighbors using a message that includes its source node ID, source node location and the potential value. Each receiving node can compute the distance from the source, based on the encoded source location and its own known location, and compute the component of the potential field due to that message.

Figure 8 is an example of this algorithm in operation. It depicts a sensor network being used to sense a cold, low lying, combustible chemical cloud as it spreads from an industrial accident. The red represents the presence of danger, the skull and crossbones are the sensor locations, and the arrows show
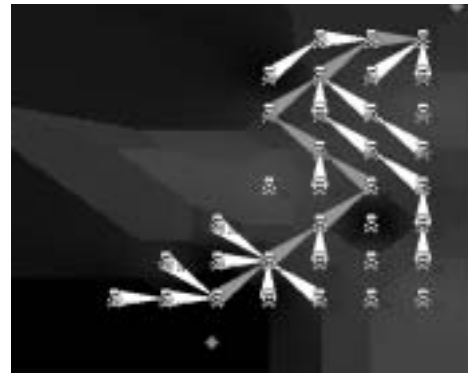

Fig. 8. A graphic display from a human guidance application, which uses a potential field algorithm to compute the safest path. The regions of brighter background represents danger due to a chemical fog, the skull and crossbones are the sensor locations, and the arrows are the safest directions to travel starting from each sensor, with an optimal path from a specific starting point to a goal highlighted down the center of the image.

the results of a safest path computation. The highlighted arrows represent the optimal path to follow from a starting location to a goal. The rest of the arrows indicate the optimal direction from other starting locations in the sensor field, which could come into play if the robot needs to deviate from the

optimal path due to an obstacle. In a large sensor network, this entire guidance map could represent a huge amount of data, which could not possibly be relayed through the low bandwidth of the sensor nodes to the robot. Thus, this serves as a good example of an application where the guidance information must reside in the sensor network to remain useful. The results of experiments using this algorithm to guide a robot are described in Kotay, Peterson, and Rus (2005).

The remainder of this section discusses a second method we call "path routing", which enables us to "embed" one or more paths in the sensor network. This method is representative of several classes of in-network path computation where the final assembly of the path must, for algorithmic or other reasons, be computed at a single node and hence distributed from that node. For efficiency we have only implemented the dissemination of the path, since we are primarily interested in the interaction between the robot and the sensor network. Sending the final complete path directly to the robot would be the most desirable way to guide the robot. However, as described above, there are reasons why this may not be possible.

3.1. Approach

The protocol is an instance of geographic routing tailored to navigation (Karp and Kung 2000). Hop-by-hop communication is used to identify the sensor nodes lying on the path. A message is broadcast which contains a list of coordinates. Each sensor that receives the message checks to determine if it lies within path width distance of a line connecting the coordinates. Sensors that belong to the path forward the path message, and those further away do not. Sensors on the path change an internal state variable and store path data, which can later be queried by the mobile node and used for navigation. Compared to flooding protocols, where all nodes receive and forward the information, the path routing protocol greatly reduces the amount of message traffic, reducing network congestion and node power consumption. It has the disadvantage of being susceptible to gaps in the sensor field, around which it cannot route if the gap cuts across the path. This can be alleviated to some extent by choosing an appropriate path width or by adding acknowledgment messages to assure the path message reaches its destination. An approach similar to greedy perimeter routing (Karp and Kung 2000) could also be used to route around obstacles. The rest of this section presents the details of our method.

A path is an array of X, Y coordinates designating waypoints along a route. A path comprises one or more sections, each of which is a set of up to 11^2 straight-line segments defined by waypoints. To establish a path, a base-station or robot sends a **Path** message. This message is 118 bytes long and its payload includes up to 12 waypoint coordinates and a path ID.

There are two phases involved in establishing an active path. First, the **Path** message must be propagated to the start of the path. Secondly, the path is activated by storing it in the sensors that lie along the path (see Figure 3). This two phase routing and distribution algorithm is summarized in Algorithm 1.

The first phase commences with a **Path** message being issued by a base-station or robot. Sensors that receive the **Path** message examine it and use the knowledge of their own location and the location of the path segments (within the message) to determine if they are on the path and within the path width defined in the message. If they are, they rebroadcast the message and set an internal flag to indicate they are on an active path. If they are not on the path, then they again use the knowledge of their own location and that of the sender (contained in the message) to determine if they are in the direction toward where the path starts, and if they are within a preset width of that direction vector (see Figure 1). If they are, they forward the message; if not, they remain silent. In this way the **Path** message is routed in the general direction of the start location of the path, without flooding the entire sensor network with messages.

In the second phase the message is routed only along the path, activating the sensors on the path. To prevent infinite loops of messages (i.e., a message bouncing back and forth from one side of the path to the other forever) each sensor keeps track of the unique ID in the path message for the last N messages it received. If a received message has been previously seen, it will be ignored. Note that multiple paths can be computed, stored, and updated by the network to match multiple robots and multiple goals. This can be easily supported by marking each robot, goal, and path pair with an ID.

A distributed motion planning protocol can run continually, perhaps in parallel with a potential field map computation, to compute, store, and update paths. Different path computation algorithms can be run as distributed protocols on top of the distributed map. For example, the safest path to the goal (which maintains the largest possible distance to each "obstacle") can be identified with a distributed protocol using dynamic programming (Li, de Rosa, and Rus 2003). The shortest path to the goal can be computed very easily by following the sensor value gradient. We are currently testing ideas on dynamic sensor-based path adaptation.

3.2. Experimental Results

In order to measure the sensor network response to computing, updating and propagating path information we have implemented the algorithms described in Section 3 on the outdoor deployed sensor network. Several different types of path were tried and the method worked reliably.

Figure 9 shows path propagation results from five different runs. Each path consists of 17 intermediate points, arranged in a U shape around the exterior of the Mote grid. The spacing

---
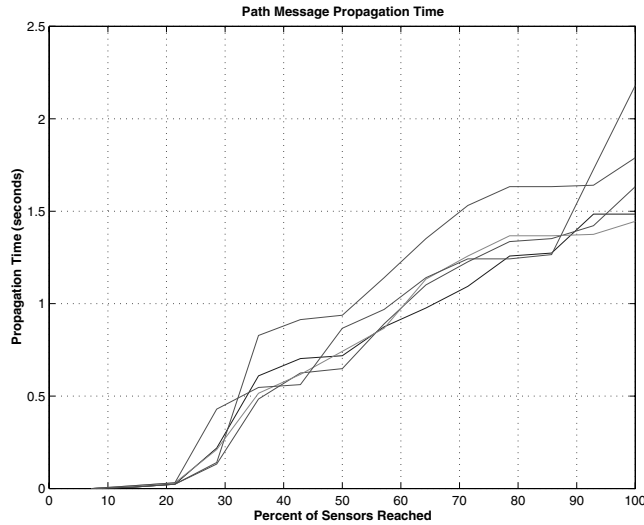
2. Limited by Mote message length.

Fig. 9. Path propagation time for five different paths over a grid of 54 Mote sensors. The *y*-axis shows the time and the *x*-axis the percentage of the sensors that are on the path and have seen the path message.

between each two Motes was 6 m, so the total path length was 96 m. The average path propagation time is 1.7 s, which translates into a speed of 56 m s^{-1}. This propagation time is very fast compared to the speed of the flying robot. We conclude that the path computation is practical for controlling the navigation of a flying robot that needs to adapt its path to changes in the environment.

For our geographic routing we observed two to six messages per sensor along the path, whereas for flooding all the sensors become involved in message forwarding, each of them receiving between 14 to 17 messages. This vector style of routing is clearly much more efficient than flooding in terms of the number of messages required.

4. Network-assisted Robot Navigation

The path stored in the sensor field by the methods just described can now be used to navigate the robot.

4.1. Approach

Similar to the way in which the **Path** message is propagated, the process has two phases: first, 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 loca-
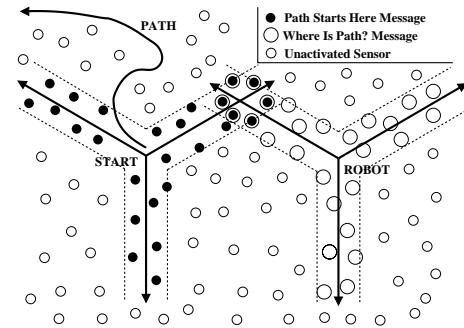

Fig. 10. The robot discovers the start of the path by sending radial messages which intersect those sent by the path head.

tion. Algorithm 2 summarizes an efficient method for guiding the robot to the path.

Algorithm 2. 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 **AnnouncePathStart** with 0 degree heading to MAXRANGE distance
>   Broadcast **AnnouncePathStart** with 120 degree heading to MAXRANGE distance
>   Broadcast **AnnouncePathStart** 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**

For the robot to find the path, first one (or all) of the sensors that know they are near the start of the path send out three messages each containing the location of the start of the path. The messages also contain a heading direction, set 120° apart,[3] a width for the vector they will travel along, and a maximum range beyond which they do not travel. The messages are forwarded out to that range in each of the three directions (see Figure 10). The sensors that forward the messages store the location of the start of the path. At some later time, the robot 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° 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

---

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

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 out to a maximum range carry messages, not the entire network. We believe this to be a general and efficient approach to finding the location of any resource the sensor field knows about. After the initialization phase which places the robot on the path, the navigation guidance algorithm summarized as Algorithm 3 is used to control the motion of the robot.

Algorithm 3. 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*

The robot starts by sending out a `QueryOnPath` message which includes the sender's ID and location. If received by a sensor on the path, it replies with a `QueryAck` message which includes the path section, some consecutive waypoints, and a sequence number indicating where these waypoints fit into the path sequence. By gathering lists of segments from multiple sensors the entire path can be assembled piece by piece as the robot moves. Paths that cross themselves allow for some fault tolerance in the robots knowledge of the path, since if the robot loses the path, it may have a future segment already stored if it has passed an intersection. Once the robot has acquired path segments from a sensor, it can then arrange them sequentially 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.

4.2. Experimental Results

This experiment was conducted indoors with the flying robot simulator. The nodes were localized, and a `Path` message was sent from the base-station to establish a path through the Mote field. The `Path` message propagated using the algorithm described in Section 3. Then the robot was turned loose in a path following mode, using the algorithm in Section 4. 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 1m width.

4.2.1. Discussion

The actual path itself was stored as perfectly precise information in these Motes and hence the robot was able to obtain precise waypoints to follow, resulting in perfect path following (within the tolerances of the robot) as shown in Figure 11. The localization accuracy only needs to be sufficient to ensure path propagation.

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 six to seven of them were not during each test, either due to defunct radios, or due to not hearing any messages for other reasons.)

It would be possible for the network to return the direction to the next node, instead of its geographic coordinate. This direction could be computed by the node when queried, or be carried as part of the path definition message. Directional information could guide a simpler robot that was not fitted with GPS and had the simple ability to follow a demanded heading.

4.2.2. Guiding Humans

The techniques we have developed for guiding robots can be readily extended to humans, and we use the human sensor network interface described in Section 1.2. The Flashlight (Figure 2) can be used to interactively define paths through the sensor network or to allow a person to follow network defined paths, using methods derived from those described in Section 4. The Flashlight was programmed to query the sensor network and receive messages in return, indicating a preferred direction towards a goal. When aligned with the most recently received direction, the Flashlight vibrator and LED activated. The directions sent by the sensors in the network were hard coded, but could just as well have been computed by the sensors or sent to the sensors using a path message. In this way, the Flashlight can be used to guide a human user along a path towards a goal, e.g., through a building towards an exit during a fire, while avoiding areas too hot for human survival. Although we found that there are problems with using magnetic compasses indoors, we measured an average directional error of 8% (30°), which was sufficiently accurate to successfully navigate a path.

5. Discussion

In the outdoor experiments, even though the Motes were fitted with external helical antennas and transmit power was set to maximum we found that the communications range was poor, not quite the 6 m Mote spacing. Indoors we had reliable communication at ranges of 10–15 m through walls. We found that this loss of communications range outdoors was due to close proximity with the ground which was fairly
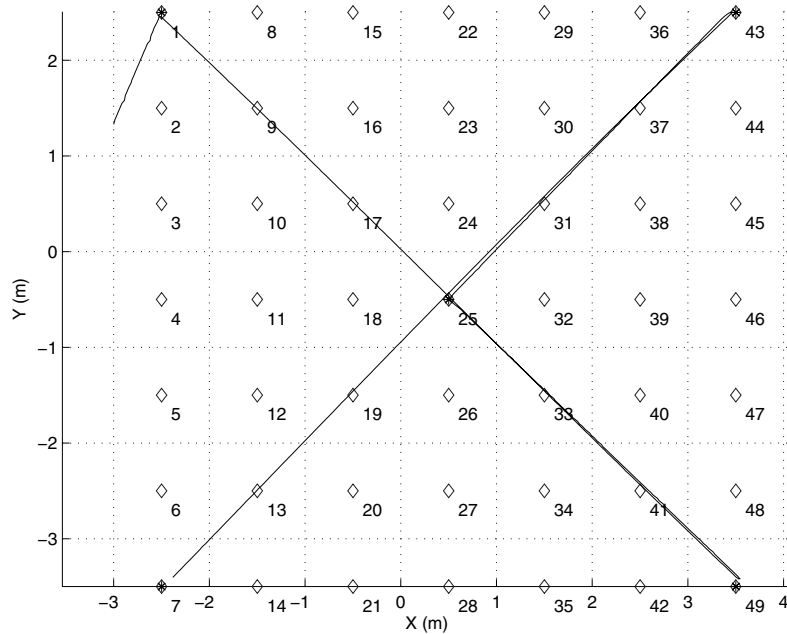
Fig. 11. Path following performance. The actual path followed by the robot is shown in black, and the asterisks indicate waypoints. The path started at node 7. See also Extension 1.

moist. We found that raising the Motes about 16 cm[4] off the ground made a significant improvement to the transmission range. We found that the ground-to-air and air-to-ground communication ranges were symmetric. However, air-to-ground communication was much longer range than Mote-to-Mote communication.

We noticed that Mote communication reliability dropped off smoothly when Motes were moving apart, but only improved stepwise for Motes moving together. Measurements of received signal strength showed this phenomenon clearly. Relative orientation of the two antennas also makes a difference as does the orientation of the helicopter, since the body of the vehicle acts as a shield for Motes behind it.

We have gained several other insights into networked robots. Data loss is common in sensor networks and has many causes, including network congestion, transmission interference, and garbled messages. We observed that the transmission range in one direction may be quite different from that of the opposite direction. Thus, the assumption that if a node receives a packet from another node, it can send back a packet, is too idealistic. Network congestion is very likely when the message rate is high. This is aggravated when nodes in close proximity try to send packets at the same time. For a sensor network, because of its small memory and simplified protocol stack, congestion is a significant problem. The uncertainty introduced by data loss, asymmetry, congestion, and

transient links is fundamental in sensor networks and should be carefully considered in developing models and algorithms for systems that involve sensor networks.

6. Conclusions

In this paper, we have presented ideas about how a network of sensors and robots can cooperate to solve important robotics problems. We identified eight forms of cooperation and in this paper have provided algorithms and experimental validation for a subset of these. As examples of this paradigm we have shown how a robot can localize sensor nodes, and how these localized nodes can navigate robots and humans through the sensorized space. We have tested these ideas in experiments with two large-scale sensor network and robot experiments involving 50 motes and two different types of flying robot. These have shown the effectiveness of geographic or vector routing, and the efficacy of using the flying robot to localize nodes. We have presented distributed algorithms used in the experiments for localization, geographic routing, path definition and incremental navigation and demonstrated them in the context of robot or human navigation.

We have only begun to explore the possibilities of sensor network and robot cooperation. In the work described, we have demonstrated the ability to load paths into the deployed sensor field and test the robot and human navigation algorithms. Future work will focus on gathering data from

---

4. This is one-half wavelength at 916 MHz, the Mote operating frequency.

robot navigation trials and demonstrating sensor-based path adaptation.

Appendix: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org.

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Network assisted navigation
2	Code	Monte Carlo simulator

Acknowledgments

References

Agre, J. and Clare, L. 2000. An integrated architecture for cooperative sensing networks. *IEEE Computer* 33:106–108.

Basagni, S., Chlamtac, I., and Syrotiuk, V. R. 1998. A distance routing algorithm for mobility (dream). *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Dallas, TX, October 25–30, pp. 76–84.

Basch, J., Guibas, L. J., and Zhang, L. 1997. Proximity problems on moving points. *Proceedings of the 13th Symposium of Computational Geometry*, Comp Geom 1997: Nice, France; the Galsyan paper appeared in the following conference: Information Processing in Sensor Networks (IPSN-2004), Berkeley, CA, pp. 344–351.

Batalin, M. and Sukhatme, G. 2005. Coverage, exploration and deployment by a mobile robot and communication network. *Journal of Robotic Systems*, Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks, 26(2):181–196 (2004).

Bulusu, N., Heidemann, J., and Estrin, D. 2001. Adaptive beacom placement. *Proceedings of the 21st Conference on Distributed Computing Systems*, Phoenix, AZ.

Buskey, G., Corke, P., Roberts, J., Ridley, P., and Wyeth, G. 2003. The CSIRO autonomous helicopter project. *Experimental Robotics VIII*, B. Siciliano and P. Dario, editors, Vol. 5 of *STAR*, Springer-Verlag, Berlin.

Chang, J.-H. and Tassiulas, L. 1999. Routing for maximum system lifetime in wireless ad hoc networks. *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL.

Chang, J.-H. and Tassiulas, L. 2000a. Energy conserving routing in wireless ad hoc networks. *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March.

Chang, J.-H. and Tassiulas, L. 2000b. Fast approximate algorithms for maximum lifetime routing in wireless ad hoc networks. *Proceedings of NETWORKING 2000, Lecture Notes in Computer Science*, Vol. 1815, Springer-Verlag, Berlin, pp. 702–713.

Chen, J. K. Y. and Hudson, R. 2002. Source localization and beamforming. *IEEE Signal Processing Magazine* 19:30–39.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. 2001. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 16–21.

Cheng, C., Riley, R., Kumar, S. P. R., and Garcia-Luna-Aceves, J. J. 1989. A loop-free extended Bellman–Ford routing protocol without bouncing effect. *Computer Communication Review* 19:224–236.

Corke, P., Peterson, R., and Rus, D. 2003. Networked robots: flying robot navigation with a sensor net. *Proceedings of the 2003 International Symposium on Robotics Research*, Siena, Italy.

Das, A., Kantor, G., Kumar, V., Pereira, G., Peterson, R., Rus, D., Singh, S., and Spletzer, J. 2003. Distributed search and rescue with robot and sensor teams. *International Conference on Field and Service Robotics*, Japan, July 14–16.

Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. 1999. Next century challenges: scalable coordination in sensor networks. *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, WA, August.

Galstyan, A., Krishnamachari, B., and Lerman, K. 2003. Distributed on-line localization in sensor networks using a moving target. *ACM SENSSYS*.

Guibas, L. 2002. Sensing, tracking, and reasoning with relations. *IEEE Signal Processing Magazine* 19:73–85.

Gupta, P. and Kumar, P. R. 1998. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming*, Birkhäuser, Boston, pp. 547–566.

Haas, Z. J. 1997. A new routing protocol for the reconfigurable wireless network. *Proceedings of the 1997 IEEE 6th International Conference on Universal Personal Communications (ICUPC'97)*, San Diego, CA, October, pp. 562–566.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. 2000. System architecture directions for network sensors. *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, November 12–15.

Hill, J., Bounadonna, P., and Culler, D. 2001. Active message communication for tiny network sensors. *INFOCOM 2001, 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, Anchorage, Alaska, April 22–26.

Johnson, D. B. and Maltz, D. A. 1996. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, T. Imielinski and H. Korth, editors, Kluwer Academic, Dordrecht, pp. 153–181.

Karlin, S. and Taylor, H. M. 1975. *A First Course in Stochastic Processes*, 2nd edn, Academic Press, New York.

Karp, B. and Kung, H. 2000. GPSR: Greedy Perimeter Stateless Routing for wireless networks. *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Boston MA, August 6–11.

Ko, Y. B. and Vaidya, N. H. 1998. Location-aided routing (LAR) in mobile ad hoc networks. *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Dallas, TX, October 25–30, pp. 66–75.

Kotay, K., Peterson, R., and Rus, D. 2005. Experiments with robots and sensor networks for mapping and navigation. *Proceedings of the 5th International Conference on Field and Service Robotics (FSR05)*, North Queensland, Australia, July.

Kotz, D., Gray, R., Nog, S., Rus, D., Chawla, S., and Cybenko, G. 1997. Agent Tcl: targeting the needs of mobile computers. *IEEE Internet Computing* 1:58–67.

Kotz, D., Newport, C., Gray, R. S., Liu, J., Yuan, Y., and Elliott, C. 2004. Experimental evaluation of wireless simulation assumptions. *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Venice, Italy, October 4–6, pp. 78–82.

Li, Q., Aslam, J., and Rus, D. 2001. On-line power-aware routing in wireless ad hoc networks. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 16–21, pp. 97–107.

Li, D., Wong, K., Hu, Y. H., and Sayeed, A. 2002. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine* 19:17–29.

Li, Q., de Rosa, M., and Rus, D. 2003. Distributed algorithms for guiding navigation across a sensor net. *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom)*, San Diego, CA, September 14–19.

Murthy, S. and Garcia-Luna-Aceves, J. J. 1996. An efficient routing protocol for wireless networks. *ACM/Baltzer Journal on Mobile Networks and Applications* Vol. MANET, pp. 183–197.

Okino, C. and Cybenko, G. 1999. Modeling and analysis of active messages in volatile networks. *Proceedings of the 37th Allerton Conference on Communication, Control, and Computing*, Monticello, IL, September.

Peterson, R. and Rus, D. 2002. Interacting with a sensor network. *Proceedings of the 2002 Australian Conference on Robotics and Automation*, Auckland, NZ, November.

Pottie, G. J. 1998. Wireless sensor networks. *IEEE Information Theory Workshop*, San Diego, CA, pp. 139–140.

Pradhan, S., Kusuma, J., and Ramchandran, K. 2002. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine* 19:51–60.

Ramanathan, R. and Hain, R. 2000. Topology control of multihop wireless networks using transmit power adjustment. *INFOCOM 2000*, Tel Aviv, Israel, March.

Rappaport, T. and Sandhu, S. 1994. Radio-wave propagation for emerging wireless personal-communication systems. *IEEE Antennas and Propagation Magazine* 36:14–24.

Simic, S. and Sastri, S. 2002. Distributed localization in wireless sensor networks. Available at http://citeseer.ist.psu.edu/correct/556794.

Simmons, R. and James, D. 2001. *Inter-Process Communication*, 3.4 edn, Carnegie-Mellon University.

Singh, S., Woo, M., and Raghavendra, C. S. 1998. Power-aware routing in mobile ad hoc networks. *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Dallas, TX, October, pp. 181–190.

Wattenhofer, R., Li, L., Bahl, P., and Wang, Y. 2001. Distrib. topology control for power efficient operation in multihop wireless ad hoc networks. *INFOCOM 2001, 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, Anchorage, Alaska, April 22–26.

Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-informed energy conservation for ad hoc routing. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 16–21.

Zhao, F., Shin, J., and Reich, J. 2002. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine* 19:61–72.