Communication-assisted Localization and Navigation for Networked Robots

Peter Corke* Ron Peterson† Daniela Rus‡

April 7, 2004

Abstract

This paper introduces the application of a sensor network to navigate a flying robot. We have developed distributed algorithms and efficient geographic routing techniques to incrementally guide one or more robots to points of interest based on sensor gradient fields, or along paths defined in terms of Cartesian coordinates. These include a distributed robot-assisted localization algorithm, a distributed communication-assisted path computation algorithm for the robot and a distributed communication-assisted navigation algorithm to guide the robot. The robot itself is an integral part of the localization process which establishes the positions of sensors which are not known a priori. The sensor network is an integral part of the computation and storage of the robot's path.

We use this system in a large-scale outdoor experiment with Mote sensors to guide an autonomous helicopter along a path encoded in the network. We also describe how a human can be guided using a simple handheld device that interfaces to this same environmental infrastructure,

1 Introduction

We wish to create more versatile information systems by using networked robots and sensors: thousands of small low-cost sensors embedded in the environment, mobile sensors, robots, and humans all interacting to cooperatively achieve tasks. This is in contrast to today's robots which are complex monolithic engineered systems that operate alone.

Recent advances have shown the possibilities for low-cost wireless sensors, with developments such as the Mica Mote [23, 24] and the single chip called "Spec" [1] along the path to the ultimate goal of smart dust. Other technologies such as AutoId will soon embed a wireless device with a globally unique identifier into every manufactured article. This leads to a paradigm shift in robotics which has traditionally used

*CSIRO ICT Centre, Australia, (e-mail: peter.corke@csiro.au).

†Dartmouth Computer Science Department, Hanover, NH 03755 USA, (e-mail: rapjr@cs.dartmouth.edu).

‡Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA 02139 USA, (e-mail: rus@csail.mit.edu).

a small number of expensive robot-borne sensors. The new model is ubiquitous sensors embedded in the environment with which the robot interacts: to deploy them, to harvest data from them, and to task them.

A robot network consists of a collection of robots distributed over some area that form an ad-hoc network. The nodes of the network may be heterogeneous and include mobile robots, mobile and static sensors, even people or animals. Each sensor is equipped with some limited memory and processing capabilities, multiple sensing modalities, and communication capabilities. 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. An ad-hoc network is a temporary wireless network formed without the aid of any established infrastructure or centralized administration. This network can support robot-robot communications, or in a first responder scenario also support human-human, human-robotic and sensor-human communications. Such systems are well-suited for tasks in extreme environments, especially when there is no computation and communication infrastructure and the environment model and the task specifications are uncertain and dynamic. For example, a collection of simple robots can locate the source of a fire or chemical leak by moving along a sensory signal gradient.

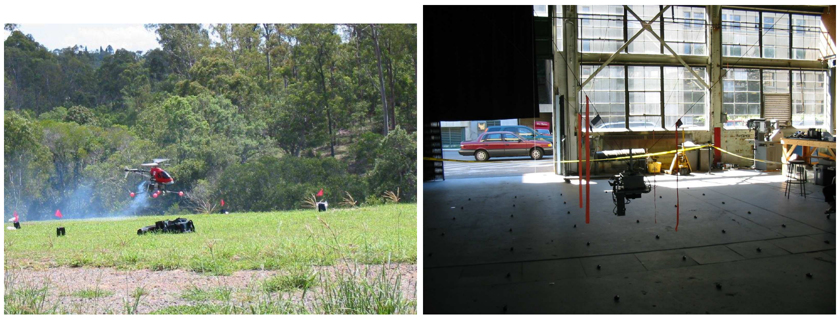


Figure 1: (Left) Helicopter in the air over the outdoor sensor network consisting of 54 Motes [23, 24]. The Motes sit on top of the dark flower pots. (Right) The experimental testbed consisting of 49 Motes on the ground and the flying robot simulator.

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 are dropped from a flying robot and they localize using GPS locations broadcast by the robot. Once localized, they sense and propagate temperature levels to compute a temperature gradient for the region. The occurrence of a new fire will 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. The sensor network can update these path 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 co-exist within the system.

Robot guidance is achieved by the interaction between the robot and a local node which has access to global state via the network. The reverse is also possible, the robot may 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. The ability to re-task 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. The capabilities of robots or people is extended through interaction with the network, extending their senses and ability to act over a massive area.

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 facilitates sensor network localization by making GPS data available to all nodes. In turn, the sensor network helps the navigation of the flying robot by providing information outside the robot's immediate sensor range. In our previous work [14] we introduce robot-assisted localization. In this paper we discuss in detail algorithms for node localization and node navigation that use communication. Our algorithms are based on efficient geographic routing methods that minimize network power consumption and radio congestion. These concepts have been experimentally validated with a physical sensor network consisting of 54 Mote sensors [23, 24] and an autonomous helicopter. Finally, we present extensions to guiding humans along safe paths.

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 [23]. Important contributions on which this work builds include [2, 7–9, 15, 16, 32, 36, 42]. Sensor network mobility issues are discussed in [5]. Other key results in controlling sensor networks include node design, routing, control of information gathering, representation of information, and in-network information processing [10,11,19,22,23,30,37,38,43–45]. Much work in sensor networks builds on results in ad-hoc networks that address the limitations of wireless networks (low bandwidth, high error rates, low power, disconnections) [3, 4, 12, 20, 21, 21, 25, 26, 28, 28, 29, 31, 33, 33, 34, 42].

The node localization problem has been previously discussed by others and usually requires estimates of inter-node distance, a difficult problem. Simić and Sastry [40] present a distributed algorithm that localizes a field of nodes in the case where a fraction of nodes are already localized. Bulusu etal. [6] propose a localization method that uses fixed beacons with known position. Galystyan etal. [17] 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. Ganesan etal. [18] show that in reality (for Rene Motes which use the same transceiver as the Mica Motes) the communications region has a complex non-circular shape and that the probability of message reception, as well as signal strength varies in a complex manner with distance [39]. These observations accord with our experimental experience. All results reported to date have been based on simulation and assume a circular radio communications region which is far from reality.

2 Navigating with a Sensor Network

Sensors sample local state information. They can perform simple local computation, store information locally or communicate it. We assume that the sensors have reliable (but not perfect) communication with nearby neighbors and non-reliable communication with the rest of the network. The sensors form an ad-hoc network. The network can be extended to include mobile nodes such as flying robots, ground robots, or humans.

We have developed and implemented a control algorithm that allows flying robots to fly along paths computed adaptively by a sensor network and communicated incrementally to the robot. The information necessary for navigation is distributed between the robot and the network. The network contains local data about the environment and can use this data to generate global maps, while the robot has information about the task. We are also able to embed a path, computed externally or by the network, into the network itself.

Our flying robot can be thought of as a mobile node in the sensor network. The flying robot is equipped with a sensor node that allows the robot to be networked to the rest of the system. The robot does not have direct access to reMote sensor data because the communication ranges are limited and there is no other infrastructure available to the robot. However, by using ad-hoc routing, navigation information that takes the entire region into account can be delivered to the robot. This data distribution is useful for applications where the path of the flying robot depends on environmental conditions. The robot's access to data measured and communicated by reMote sensors via the network allows it to respond quickly to distant events and adjust its actions accordingly.

The problem can be formulated as follows. A sensor network is dispersed over a large geographical area (see [13] for a solution to deployment). A flying robot is tasked to travel along a path across this area to reach multiple goal locations that may change dynamically. The sensor network computes the goals and the best path that visits each goal adaptively. Note that multiple robots can be guided to different goals at the same time by the system, along different paths. The robot, which is equipped with a GPS receiver, is also used to initially localize the nodes.

Realizing this type of cooperative control of a mobile robot requires three capabilities. The nodes in the sensor network need location information in order to support path computation. The nodes in the network must be able to efficiently compute, modify, and store a path for the mobile robot. The mobile robot must be able to interact with the sensor network to receive the path and to respond to changes in the path. The following sections detail the algorithms for these three capabilities.

2.1 Robot-assisted Localization

In [14] 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. 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 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 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.

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
  $s_{max} = s_i$
  $\hat{p} = p_i$
```

mean Assume 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 $\hat{p}_i = \text{median}(p_{1\ldots i})$

constraint Consider each received position as a constraint [17] 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.

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 deg 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 region. 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 bf median has the disadvantage of needing to store all messages which may be problematic on memory limited hardware.

Once we have the ability to localize deployed sensors we are able to employ efficient routing techniques such as geographic routing, which increases the value and usefulness of sensed data by tagging events to geographic location, as well as using the network to guide a robot. These concepts are discussed in the following sections.

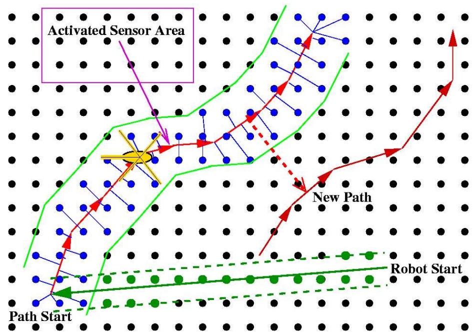


Figure 2: A sensor network with a path marked by sensor nodes. In response to an environment trigger, the sensor network computes a new path for the helicopter and an intermediate path to guide the helicopter to the new path.

2.2 Communication-assisted Path Computation

Methods to guide the robot using a sensor network fall into two main categories. Firstly the sensor network with localized nodes can monitor the environment and encode a map of the environment in sensor space as described in [32]. 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

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.
```

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.

The remainder of this section discusses a second method we call *Path Routing* which enables us to "embed" one or mores paths adaptively in the sensor network. The protocol is an instance of geographic routing tailored to navigation [27]. 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 *pathwidth* distance of a line connecting the coordinates. Sensors that belong to the path forward the path message, 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 [27] 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[1] straight line segments defined by waypoints. The waypoints are application specific and could be set by a human, computed by a robot, or computed by the sensor network. 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. Firstly, 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 2). 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

[1]Limited by Mote message length.

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 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
```

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 [32]. 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.

2.3 Communication-assisted Robot Navigation

The *path* stored in the sensor field can be used to navigate the robot. Similar to the way in which the **Path** message is propagated, the process 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 2 summarizes an efficient method for guiding the robot to the path.

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

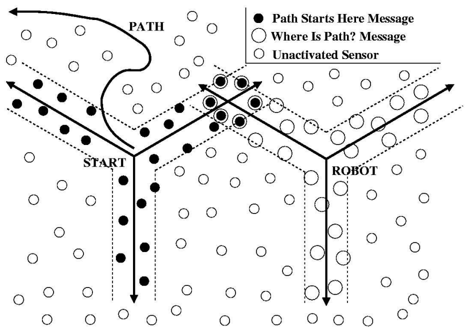


Figure 3: The robot discovers the start of the path by sending radial messages which intersect those sent by the path head.

of the path. The messages also contain a heading direction, set 120 deg apart[2], 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 3. 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 deg 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 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.

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.

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

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
```

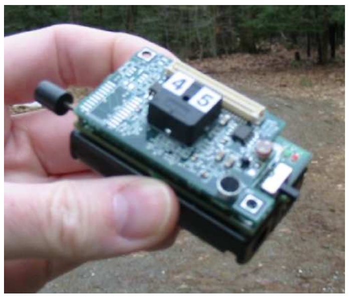


Figure 4: A Mica Mote with sensor board and long range antenna.

3 Experiments

3.1 Experimental setup

The 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 [23,24], see Figures 1 and 4. 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 (50Kbits/sec, nominal 30m range), a UART and a 4Mbit 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.

The Autonomous Robot The CSIRO helicopter, see Figure 1, is a hobby type (60 class) JR Ergo, which has a limited, 5kg, payload capability. 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 vision system, implemented in software, provides height relative to the ground and speed from optical flow between consecutive frames at 5Hz. A flight computer located in the nose acts as the interface between the helicopter and the control computer, allowing the computer to monitor or take over any servo channel.

The control computer is an 800MHz P3 with solid-state disks running the LynxOS operating system. It is responsible for running the vision software, the control loops and data logging. A 1Hz differential GPS receiver and a Proxim radio ethernet card are also fitted. A Mote is fitted to the nose of the helicopter and functions as a base-station. It communicates over a serial link with the control computer which runs application software to interact with the sensor network on the ground. For the localization experiments it broadcasts the helicopter's differential GPS position once per second.

Experimentalal sites In March 2003 we conducted outdoor experiments with the robot helicopter and 54 Mica Motes, see Figure 1, at the CSIRO site in Brisbane. The Motes were placed at the nodes of a 6m 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 3.5.1. A base-station Mote connected to a laptop was used to control the Mote network. Figure 5 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 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 49 Mica Motes [23, 24] and a flying robot simulator. For this experiment, 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 [41]. 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 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, see Figure 6(a), where the diamonds represent the surveyed positions of the Motes.

The Flashlight sensor interface [35], see Section 3.5.2, was used to adjust the RF power of the sensors in the grid to an optimal level for communication with the robot, essentially a trial-and-error adjustment, gradually incrementing the Mote power until the robot was getting good communications.

3.2 Localization Results

In this section we compare empirically the performance of the five[3] different approaches to localization introduced in Section 2.1 using data acquired during experiments. The error between estimated and actual Mote coordinate for each of the algorithms is shown in Figure 5. The results have been computed offline using GPS coordinates obtained from the actual helicopter path shown in Figure 5. The parameters used were $d = 20$ and $s_{min} = 470$. We can see that the mean and weighted mean are biased, particular in the Easting direction, due to the path taken by the helicopter and/or the lobe shape of the Mote antenna. The method **strongest** is simple but has high residual error. The **median** does not perform significantly better than the **mean** or **wmean** estimates. The **constraint** method was arguably the best performer and is computationally cheap, though it is sensitive to the choice of d.

The errors shown should be considered with respect to the accuracy of differential GPS itself which is of the order of several metres. Achievable localization accuracy is of the order of one half the grid spacing which is more than sufficient to enable the geographic routing strategies discussed above. We note that the methods do not require a range estimate derived from signal strength, a difficult inverse problem [39], nor make any assumption about the size or shape of the radio communications region.

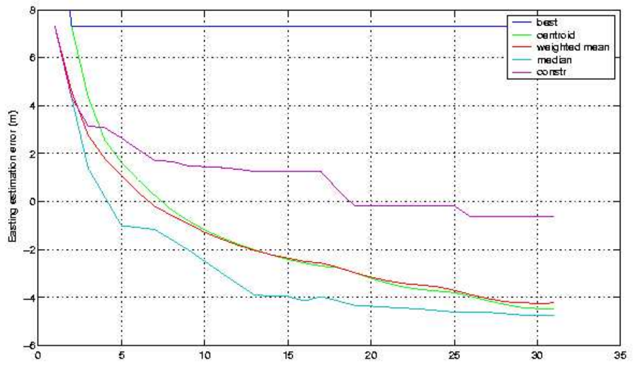
In the experiments with the flying robot simulator at CMU the robot followed a serpentine path, see Figure 6(a). Once per second the flying computer obtained its current coordinate from the control computer using IPC over the 802.11 link, and broadcast this via the onboard base-stationstation Mote. Each ground Mote recorded all the X,Y broadcasts it received and used the **mean** method to estimate its location. Figure 6(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 2 and 16 broadcasts each as can be seen in Figure 6(b) with a median value of 10. Figure6(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 later download and analysis. Figure 7(a) compares the true and estimated Mote locations. We can see a general bias inward and this would be expected given the the bias in the direction from which broadcasts were received. Figure 7(b) shows a histogram of the error magnitudes and indicates a maximum value of 1.4m and a median of 0.6m which is, again, approximately half the grid spacing which we achieved with the real helicopter and differential GPS [14].

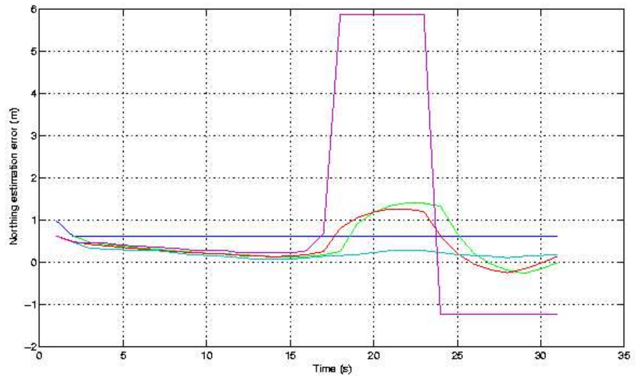
3.3 Challenges with Distributed Localization

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

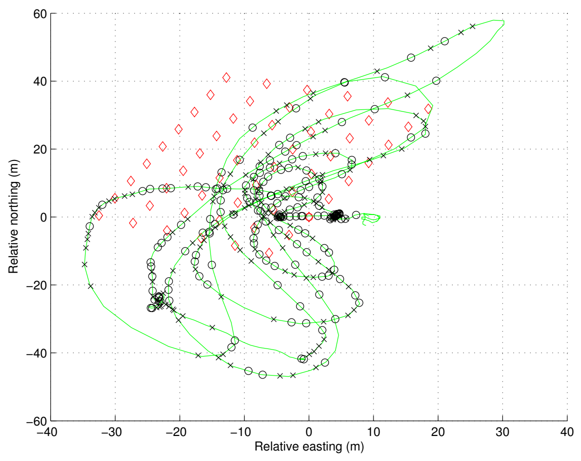
[3]The **bound** algorithm was developed subsequently.



(a)

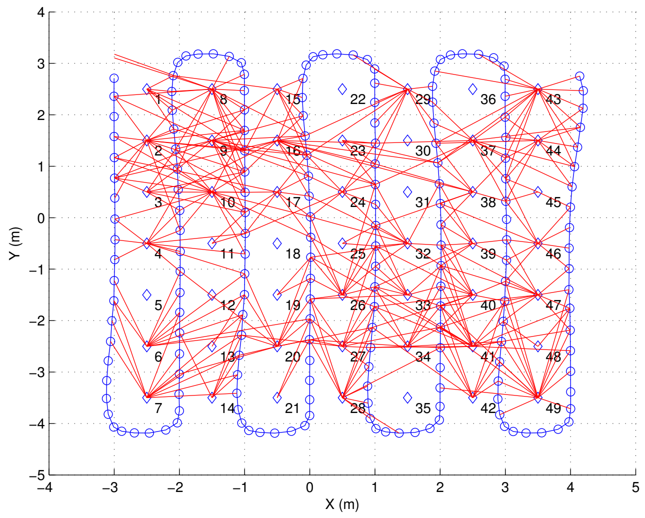


(b)

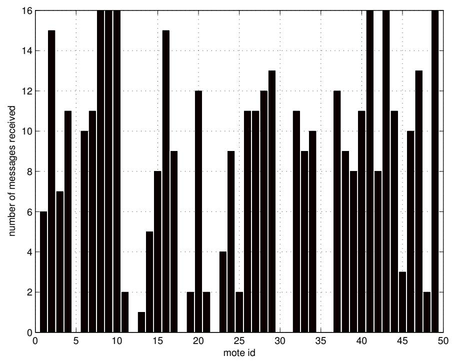


(c)

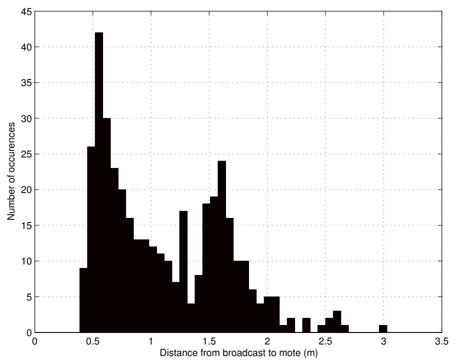
Figure 5: Results of offline localization by GPS, evolution of different estimates with time for our 5 localization methods. Error in the Easting (a) and Northing (b) directions are shown. (c) The helicopter path is shown with GPS reception marked: **o** denotes a good packet and **x** a bad packet.



(a)

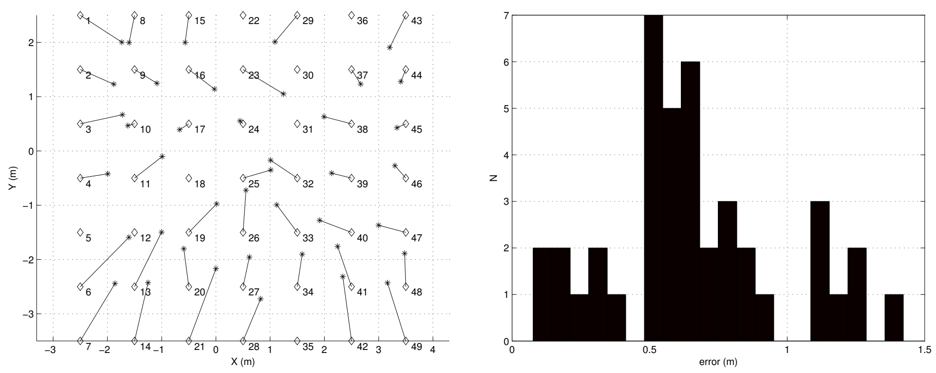


(b)



(c)

Figure 6: 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.



(a) (b)

Figure 7: Localization performance using centroid method. (a) Actual (◇) and estimated (*) location. (b) Histogram of error vector length.

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 6(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 improve the second problem include:

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 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. Decreasing the size of 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 in which we vary the rate at which the robot broadcasts its position, and the radio reception range. 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 path dependence problem 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 100×100m. 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 mean inter-node spacing is 17m. 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 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 **constr** method we set $d = 20$. The mean and maximum localization error statistics for all the nodes is 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 8 shows some of the results. 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, and the methods **constr** and **bound** perform identically since the actual and assumed transmit radii are equal.

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** and **bound** also behave well.

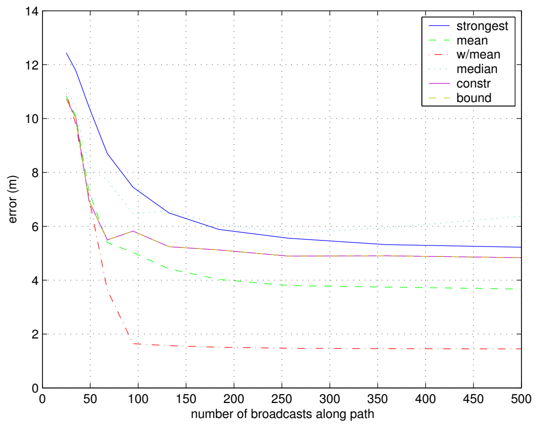
We plan to extend these numerical experiments to include stochastic packet reception models and non symmetric radio reception models.

3.4 Path Routing Results

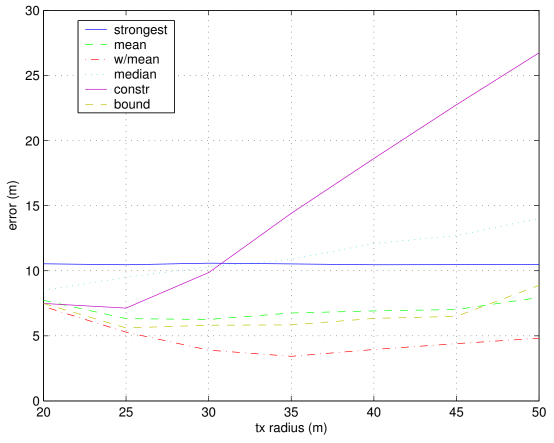
In order to measure the sensor network response to computing, updating and propagating path information we have implemented the algorithms described in Section 2.2 on the deployed sensor network. Several different types of path have been tried and the method works reliably.

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 between each two Motes was 6 meters so the total path length was 96 meters. The average path propagation time is 1.7 seconds which translates into a speed of 56 m/sec. 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 2 to 6 messages per sensor along the path, whereas for flooding *all* the sensors become involved in message forwarding, each of



(a)



(b)

Figure 8: Mean localization error from the Monte Carlo study using the six methods of Section 2.1. (a) Effect of varying the broadcast interval (transmit range = 20m). (b) Effect of varying the transmission radius (50 broadcasts along path).

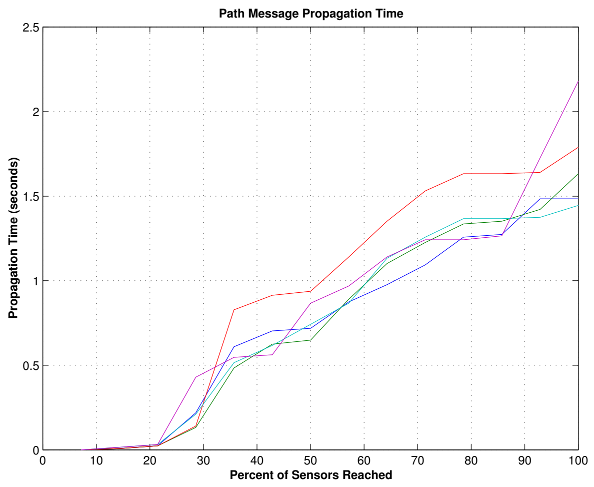


Figure 9: Path propagation time for 5 different paths over a grid of 54 Mote sensors. The y axis shows the time and the x axis the percentage of the sensors that are on the path and have seen the path message.

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.

3.5 Navigation Results

Once localized, a **`Path`** message was sent from the basestation to establish a path through the Mote field. The **`Path`** message propagated using the algorithm described in Section 2.2. Then the robot was turned loose in a path following mode, using the algorithm in Section 2.3. 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. 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 10. 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 6–7 of them were not during each test, either due to defunct radios, or due to not hearing any messages for other reasons.)

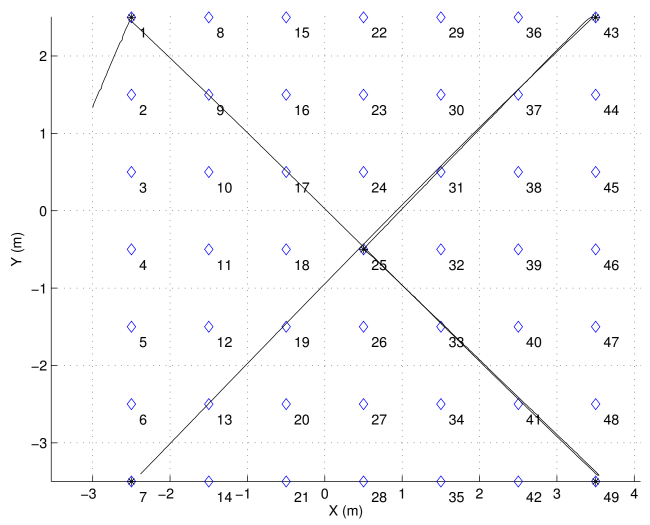


Figure 10: Path following performance. The actual path followed by the robot is shown in black, and the asterisks indicate waypoints. The path started at node 7.

3.5.1 Lessons Learned

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 6m Mote spacing. Indoors we had reliable communication at ranges of 10 to 15m through walls. We found that this loss of communications range outdoors was due to close proximity with the ground which was fairly moist. We found that raising the Motes about 16cm[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-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

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

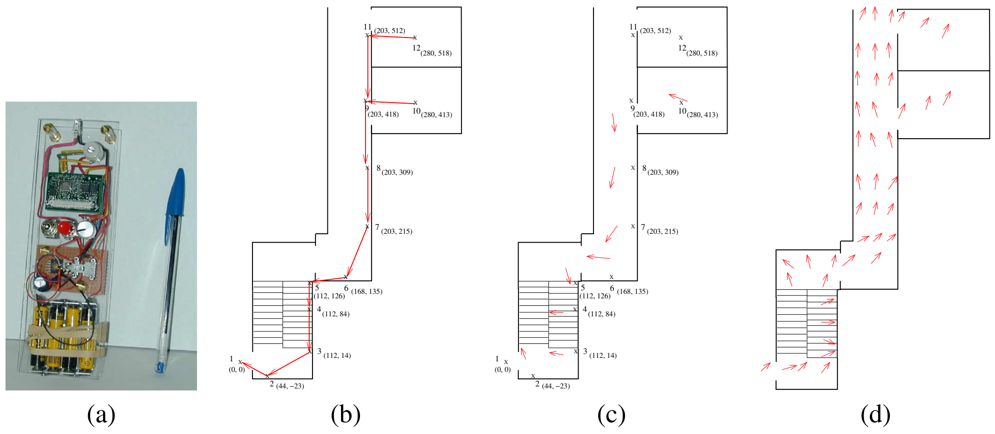


Figure 11: (a) The Flashlight device. (b) The guidance path programmed into the motes. (c) The guidance results. (d) Map of magnetic North in hallway.

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.

3.5.2 Extension to Guiding Humans

The techniques we have developed for guiding robots can be extended to humans, but we need some interface between human and the sensor network. The *sensory Flashlight*, see Figure 11(a), is a hand-held device which uses the metaphor of a flashlight to provide this connection. When pointed in a specific direction, the Flashlight collects information from all the sensors located in that direction and provides its user with feedback.

The Flashlight consists of an electronic analog compass, alert LED, pager vibrator, a 3 position mode switch, a power switch, a range potentiometer, some power conditioning circuitry, a microcontroller based CPU, and an RF transceiver. The processing and RF communication components of the Flashlight and the sensor network are Berkeley Motes [24]. The potentiometer is used to set the detection range. The electronic compass supplies heading data, indicating the pointed direction of the device. When the user points the Flashlight in a direction, if sensor reports of the selected type are received from any sensors in that direction, a silent vibrating alarm activates and the LED lights. The vibration amplitude can be used to encode how far (in number of hops or range) was the sensor that triggered. The device can also issue commands to the sensors in the direction it is pointing, causing sensors at a specified range to activate/deactivate.

In an experiment on human guidance we deployed 12 Mote sensors along corridors in our building and used the Flashlight and a modification of the path guidance approach presented above to guide a human user out of the building. Since the Flashlight only knows its orientation and not its location, the path data consisted only of compass directions. Figure 11(b) shows a map of the directions programmed into each mote. The Flashlight interacted with sensors to compute the next direction of movement toward the exit. For each interaction, the user scanned the flashlight from side to side until the Flashlight indicated the preferred direction. The user then walked in that direction to the next sensor and repeated the process. At each scan we recorded the correct direction and the direction detected by the Flashlight. Figure 11(c) shows the resulting guidance the Flashlight provided the user in finding an exit to the building. The directional error was 8% (or 30 degrees) on average and was mostly due to the variation in the magnetic field in the building as shown in Figure 11(d). Note the large magnetic deviation on the stairs, caused by the presence of metal handrailings and balusters. However, because the corridors and office doorways are wide, and the sensors sufficiently dense, the exit was identified successfully. An interesting question is how dense should the sensors be, given the feedback accuracy. Future work will focus on improving directional and positional accuracy and addressing how to cope with sensor signals received from the other side of walls and from floors above and below.

4 Conclusions

We have described a sensor network and developed novel algorithms that provide guidance information to robot or human users. Such a network greatly extends the sensory reach of an individual robot or human and provides for many different modes of navigation. We have described a networked approach to robot navigation that allows the robot to respond to remotely sensed data and adapt its heading in response to it. A sensor network and mobile robot cooperate to control the motion of the robot. The robot has task information and the sensor network contains the environment data in the form of a distributed map. The sensor nodes cooperate to update the map in response to changes, and to transmit these changes to the robot control in the form of path updates. The interaction is also bidirectional. The robot is able to provide information to the network and we have demonstrated the power of this in the task of node localization.

We have implemented the navigation protocols on a network of 54 Mote sensors in a large-scale outdoor setting, and tested aspects of helicopter and sensor network interaction. Experiments have shown the effectiveness of geographic or vector routing, and the efficacy of using the flying robot to localize nodes. Various localization algorithms were compared using experimental data. We were able to load paths into the deployed sensor field and manually test the robot and human navigation algorithms. Future work will focus on gathering data from robot navigation trials and demonstrating sensor-based path adaptation.

Acknowledgments

This work is a collaborative project between the Dartmouth Robotics Laboratory, the Rus Laboratory at MIT and the CSIRO Robotics & Automation team. The authors would like to thank the rest of the CSIRO helicopter team: Jonathan Roberts, Gregg Buskey, Srikanth Saripalli (University of Southern California), Graeme Winstanley, Leslie Overs, Pavan Sikka, Elliot Duff, Matthew Dunbabin, Stuart Wolfe, Stephen Brosnan, and Craig Worthington, and our pilot Fred Proos. The authors also thank Sanjiv Singh for facilitating the CMU experiments. Support for this work was provided through the Office of Domestic Preparednesss and the Institute for Security Technology Studies at Dartmouth, NSF awards 0225446, EIA-9901589, IIS-9818299, and IIS-99R12193, ONR award N00014-01-1-0675, the Darpa TASK program and MIT project Oxygen. We are grateful for it.

References

[1] http://robotics.eecs.berkeley.edu/ pister/smartdust/.

[2] Jon Agre and Loren Clare. An integrated architeture for cooperative sensing networks. *Computer*, pages 106 – 108, May 2000.

[3] S. Basagni, I. Chlamtac, and V. R. Syrotiuk. A distance routing algorithm for mobility (dream). In *Proceedings of ACM/IEEE MOBICOM'98*, pages 76 – 84, 1998.

[4] Julien Basch, Leonidas J. Guibas, and Li Zhang. Proximity problems on moving points. In *the 13th Symposium of Computational Geometry*, pages 344–351, 1997.

[5] M. Batalin and G. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Journal of Robotic Systems*, (to appear), 2004.

[6] N. Bulusu, J. Heidemann, and D. Estrin. Adaptive beacom placement. In *Proceedings of the* 21^{st} *Conference on Distributed Computing Systems*, Phoenix, AZ, 2001.

[7] Jae-Hwan Chang and Leandros Tassiulas. Routing for maximum system lifetime in wireless ad-hoc networks. In *Proceedings of 37-th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, 1999.

[8] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proc. IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.

[9] Jae-Hwan Chang and Leandros Tassiulas. Fast approximate algorithms for maximum lifetime routing in wireless ad-hoc networks. In *NETWORKING 2000, Proc. Lecture Notes in Computer Science, Vol. 1815*, pages 702–713, May 2000.

[10] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *7th Annual Int. Conf. Mobile Computing and Networking 2001*, Rome, Italy, July 2001.

[11] J.C.and Kung Yao Chen and R.E. Hudson. Source localization and beamforming. *IEEE Signal Processing Magazine*, 19(2):30–39, March 2002.

[12] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. A loop-free extended bellman-ford routing protocol without bouncing effect. *Computer Communication Review*, 19(4):224 –236, September 1989.

[13] P. Corke, S. Hrabarz, R. Peterson, D. Rus, S. Saripalliz, and G. Sukhatme. Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In *The IEEE 2004 International Conference on Robotics and Automation*, 2004.

[14] 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.

[15] A. Das, G. Kantor, V. Kumar, G. Pereira, R. Peterson, D. Rus, S. Singh, and J. Spletzer. Distributed search and rescue with robot and sensor teams. In *To appear in Field and Service Robotics*, Japan, July 2003.

[16] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM MobiCom 99*, Seattle, USA, August 1999.

[17] A. Galstyan, B. Krishnamachari, and K. Lerman. Distributed online localization in sensor networks using a moving target. submitted to 2003 acm senssys.

[18] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, , and S. Wicker. Complex behavior at scale: An experimental study of lowpower wireless sensor networks. *UCLA Computer Science*, (UCLA/CSD-TR 02-0013), July 2002.

[19] L.J. Guibas. Sensing, tracking, and reasoning with relations. *IEEE Signal Processing Magazine*, 19(2):73–85, March 2002.

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

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

[22] W. Rabiner Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient routing protocols for wireless microsensor networks. In *Hawaii International Conference on System Sciences (HICSS '00)*, Jan. 2000.

[23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, 2000.

[24] Jason Hill, Philip Bounadonna, and David Culler. Active message communication for tiny network sensors. In *INFOCOM*, 2001.

[25] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153 –181. Kluwer Academic Publishers, 1996.

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

[27] B. Karp and H.T. Kung. GPSR: Greedy Perimeter Stateless Routing for wireless networks. In *Proceedings of MobiCom 2000*, Aug. 2000.

[28] Y. B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of ACM/IEEE MOBICOM'98*, pages 66 – 75, 1998.

[29] David Kotz, Robert Gray, Saurab Nog, Daniela Rus, Sumit Chawla, and George Cybenko. Agent Tcl: Targeting the needs of mobile computers. *IEEE Internet Computing*, 1(4):58–67, July/August 1997.

[30] D. Li, K.D. Wong, Yu Hen Hu, and A.M. Sayeed. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine*, 19(2):17–29, March 2002.

[31] Q. Li, J. Aslam, and D. Rus. Online power-aware routing in wireless ad-hoc networks. In *MOBICOM*, pages 97–107, Rome, July 2001.

[32] Qun Li, Michael de Rosa, and Daniela Rus. Distributed algorithms for guiding navigation across a sensor net. In *Proceedings of MobiCom 2003*, 2003.

[33] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM/Baltzer Journal on Mobile Networks and Applications*, MANET(1,2):183 –197, October 1996.

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

[35] 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.

[36] Gregory J. Pottie. Wireless sensor networks. In *IEEE Information Theory Workshop*, pages 139–140, 1998.

[37] S.S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, 19(2):51–60, March 2002.

[38] Ram Ramanathan and Regina Hain. Topology control of multihop wireless networks using transmit power adjustment. In *INFOCOM*, 2000.

[39] T.S. Rappaport and S. Sandhu. Radio-wave propagation for emerging wireless personal-communication systems. *IEEE Antennas and Propagation Magazine*, 36(5):14–24, October 1994.

[40] S. Simic and S. Sastri. Distributed localization in wireless sensor networks, Available at citeseer.nj.nec.com/464015.html, 2002.

[41] Reid Simmons and Dale James. *Inter-Process Communication*. Carnegie-Mellon Univeristy, 3.4 edition, February 2001.

[42] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile adhoc networks. In *Proc. of Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 181–190, Dallas, TX, Oct. 1998.

[43] R. Wattenhofer, L. Li, P. Bahl, and Y. Wang. Distrib. topology control for power efficient operation in multihop wireless ad hoc networks. In *INFOCOM 2001*.

[44] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MOBICOM 2001*, July 2001.

[45] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.