# Networked Robots: Flying Robot Navigation using a Sensor Net

Peter Corke* Ron Peterson† Daniela Rus‡

April 18, 2003

**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. The robot itself is an integral part of the localization process which establishes the positions of sensors which are not known a priori.

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. A simple handheld device, using this same environmental infrastructure, is used to guide humans.

## 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 [10, 11] along the path to the ultimate goal of smart dust recently implemented on a single chip called Spec [1]. Other technologies such as AutoId will soon embed in every manufactured article a wireless device with a globally unique identifier. This leads to a paradigm shift in robotics which has traditionally used 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. The sensors may be static or mobile.

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. 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 and human-robotic 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.

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 while also providing a cheap distance measure. For example, consider dispersing a sensor network over a large forest to monitor forest fires. The sensors are dropped with a flying robot and they can localize using GPS locations that are beamed down from the robot. Once localized, they sense and propagate

*CSIRO Manufacturing & Infrastructure Technology, Australia, (e-mail: peter.corke@csiro.au).
†Dartmouth Computer Science Department, Hanover, NH 03755 USA, (e-mail: rapjr@cs.dartmouth.edu).
‡Dartmouth Computer Science Department, Hanover, NH 03755 USA, (e-mail: rus@cs.dartmouth.edu).


Figure 1: (Left) Helicopter in the air over the sensor network consisting of 54 Motes [10, 11]. The Motes sit on top of the dark flower pots. (Center)Helicopter. (Right) 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.

temperature levels to compute a temperature gradient for the region. The occurrence of a new fire will be signaled throughout the network automatically. In addition, the sensor network can also compute a shortest path to the fire, and safe paths to exit for people in the area. The sensor network can use this information to guide robots to the location of the fire along the path computed by the network, which may change 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 focus on robot networks, where each node is capable of sensing and communication, and some of the nodes are capable of actuation. We build on our previous work in networked robots [7, 14, 15] and controlling flying robots [6] and describe the navigation of a flying robot with a massively distributed network.

More specifically, we build on important previous work in sensor networks [2–5, 8], ad-hoc networks [10–12, 16, 17], and robotics [7, 13] and examine in more detail robot networks that provide directions to a moving user (a robot or a human). We examine the networked interaction between the moving user and the rest of the system that includes using the mobile node to localize the sensors, computing distributed maps and paths across the network, and using the network to guide a flying robot. We present efficient geographic routing techniques which minimize network power consumption and radio congestion. This requires node localization and we present for the first time experimental results for localization using a flying robot. These concepts have been experimentally validated with a physical sensor network consisting of 54 Mote sensors [10, 11] and an autonomous helicopter.

## 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 (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 control 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.

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 detected and communicated by remote sensors via networking allows it to respond quickly to far away events by adjusting its flying direction.

The problem can be formulated as follows. A sensor network is dispersed over a large geographical area. 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.

The algorithm that realizes this type of cooperative control of a mobile robot requires three modules. 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 Node Localization

A critical aspect of our approach is that the nodes are localized — for efficient routing and for knowing the location of the sensed events. For a large sensor network this requirement could be limiting since it would be impractical (for reasons of cost and power consumption) for each node to have GPS capability. Here we describe how this can be achieved by a differential GPS equipped flying robot interacting with the sensor network during a post-deployment initialization phase.

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. The flying robot can sweep across the area of the sensor network, for example along a random path or a path defining a grid, beaming down GPS coordinates.

The node localization problem has been previously discussed by others. Simić and Sastry [2] present a distributed algorithm that localizes a field of nodes in the case where a fraction of nodes are already localized. Bulusu etal. [5] propose a localization method that uses fixed beacons with known position. Galystyan etal. [9] described a contraint-based method whereby an individual node refines its position estimate based on location broadcasts from a moving agent. All results reported to date have been based on simulation and assume a circular radio communications region which is far from reality.

We have developed five simple approaches to localization that do not require inter node communications. Such constraints may improve localization accuracy but at the expense of increased power consumption and network congestion. In Section 3.2 we show empirically that these algorithms are practical and compare them using experimental data.

The flying robot broadcasts its position $p_i = (x_i, y_i)$ which is received with strength $s_i$. Each sensor listens for GPS broadcasts from the helicopter and improves its location estimate over time using one of the following algorithms.

1. Take the strongest received message so far, as the best estimate of node position.
   **if** $s_i > s_{max}$ **then**
   $s_{max} = s_i$
   $\hat{p} = p_i$
2. Take the mean of the received position as the estimate $\hat{p}_i = \frac{\Sigma_i p_i}{i}$
3. Take the signal strength weighted mean of the received position as the estimate $\hat{p}_i = \frac{\Sigma_i s_i p_i}{\Sigma_i s_i}$
4. Take the median of received position as the estimate $\hat{p}_i = \text{median}(p_{1\cdots i})$
5. Consider each received position as a constraint [9] on the node position which is considered to lie within the rectangular region $Q$. $Q_i = Q_{i-1} \cap [x_i - d, x_i + d] \times [y_i - d, y_i + d]$ At each step we constrain the node

to lie in the intersection of its current region, $Q_{i-1}$, and a square region of side length $2d$ centered on the GPS transmission. The position estimate of the node is taken as the centroid of the region $Q_i$. The parameter $d$ should reflect the size of the radio communications region.

Note that algorithms 2, 3 and 4 can be modified so that the estimate is only updated when $s_i > s_{min}$ which artificially reduces the size of the radio communications region.

## 2.2 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.
```

A sensor network with localized nodes can monitor the environment and encode a map of the environment in sensor spaces as described in [14]. Areas of the sensor network where sensors have triggered events can be represented as obstacles. The map is a distributed representation of these obstacles. This is not an accurate geometric map. The nodes in the network provide some information about how far from the event each node is. If the sensors are uniformly distributed, *the smallest number of communication hops to a sensor that triggers "yes" to the event* is a measure of the distance. Such a map can be constructed incrementally and adaptively as an artificial potential field using hop-by-hop communication. The "obstacles" correspond to events and have repulsing values and the goal has an attracting value. The potential field is computed in the following way. Each node whose sensor triggers an "event" diffuses the information about the event to its neighbors in a message that includes its source node id, the potential value, and the number of hops from the source of the message to the current node. This message is used to update the potential value at the current node. The node then broadcasts a message with its new potential value and number of hops to its neighbors.

The rest of the section details a protocol called Path routing, which is an instance of geographic routing tailored to navigation [12]. It uses hop by hop communication to identify the sensor nodes on the path. A message is broadcast which contains a list of coordinates. Each sensor that receives it does a computation to determine if it is within

*pathwidth* distance of a line connecting the coordinates, thus the sensors must have knowledge of their location. Good localization of the sensors has benefits other than in routing; it can greatly increase the value and usefulness of the data sensed and collected by the sensors (see section 2.1.) Sensors that are on the path forward the path message; those further away do not. Sensors on the path change an internal state variable indicating they are on a path and store only those segments of the path that they are on. This information 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, which also leads to reductions in power use. 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 [12] could also be used to route around obstacles. The rest of this section presents the details.

A *path* is an array of X,Y coordinates designating way points 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 way points. 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 way point coordinates and a path id.

There are two phases involved in establishing an active path. One is to get the Path message to the area where the path starts, the other is to activate the path by storing it in the sensors that lay along the path (see Figure 1(Right)). This two phase routing and distribution algorithm is summarized in Algorithm 1.

Nearby sensors that hear the Path message examine it and use the knowledge of their own location and the location of the path segments (in 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 location and the location of the sender (contained in the message) to determine if they are in the direction of where the path starts and if they are within a preset width of that direction vector (see Figure 1(Right)). 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.

When the message reaches the path it is routed only along the path, activating the sensors on the path. To prevent infinite loops of messages (i.e., a message back and forth from one side of the path to the other forever since it always gets forwarded toward the path) each sensor keeps track of the unique ID in the path message for the last N messages it received. If a message is received that has already been processed before, it is 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 runs continuously in parallel with the 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 [14]. The shortest path to the goal can be computed very easily by following the sensor value gradient. We are currently testing ideas on path adaptation based on changes detected by the sensors.

## 2.3 Robot Navigation

The *path* stored in the sensor field can be used to navigate the robot. Similar to the way 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.

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 that contain the location of the start of the path. The messages also each contain a heading, set 120 degrees apart[2], a width for the vector they will travel along, and a maximum range beyond which they are not intended to travel. The messages are forwarded out to that range in each of the three directions. The sensors that forward the

[1] Limited by Mote message length.

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

**Algorithm 2** The QueryPath algorithm for robot guidance.

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

messages store the location of the start of the path. The robot at some later time sends out the same sort of messages in three directions. If the robot and path start are in range of each others messages, the message paths will cross (due to using a 120 degree dispersal angle.) The sensor(s) at the crossing will have a stored location for the start of the path and a location for the robot and can send a directional message (perhaps with a gradually increasing width since the robot may have moved slightly) back to the robot telling it where the start of the path is. In this way only the sensors along specific lines out to a max range carry messages instead of the entire network. This might be a good general approach to finding the nearest location of any resource the sensor field knows about. After the initialization phase that places the robot on the path, the navigation guidance algorithm summarized as Algorithm 2 is used to control the motion direction of the robot.

The robot starts by 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 way points, and an indication of where these way points fit into the path. By gathering lists of segments from multiple sensors the entire path can be assembled 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 of it already stored if it has passed an intersection. Once the robot has acquired path segments from a sensor, it can then arrange them in order of precedence and follow them in order. Thus the path itself is independent of the sensor's own location and can be specified to any level of precision needed.

# 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 [10, 11], see Figures 1 and 3(Center). 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 100ft 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 operating system. The sensors are currently programmed to react to sudden increases in light and temperature but other sensory modes are possible. The sensors need to know their location coordinates and are localized using the GPS localization algorithm in Section 2.1, but the coordinates can also be provided to each sensor from a base-station.

**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 (visible in Figure 1) 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.

**Experimental site** In March 2003 we conducted outdoor experiments with a robot helicopter (equipped with a Mote) and 54 Mica Motes (see Figures 1 and 3) 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 dramatically reduced when they were placed on damp soil. Subsequently we placed them on inverted flower pots which kept them one wavelength above ground and improved the signal quality significantly. A base-station Mote connected to a laptop was used to control the Mote network.

## 3.2 Localization Results

In this section we compare empirically the performance of the five 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 2(Left). The results have been computed offline using GPS coordinates obtained from the actual helicopter path shown in Figure 2(Right) 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 or the lobe shape of the Mote antenna. The method, 'best', is simple but has high residual error. The median does not perform significantly better than the mean estimates, and would be problematic to compute with small amounts of memory. The constraint based method is arguably the best performer and is computationally cheap, though it is sensitive to the choice of $d$. The error should be considered with respect to the accuracy of differential GPS itself which is of the order of several metres.

We note that the methods do not require a range estimate derived from signal strength, a difficult inverse problem, nor make any assumption about the size or shape of the radio communications region.

## 3.3 Path Computation 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. Figure 3 shows the 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 as compared to the moving 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 them receiving between 14 to 17 messages. This vector style of routing is much more efficient in number of messages, and the percentage of sensors involved in message forwarding becomes less and less as the size of the entire network grows.

## 3.4 Navigation Results

Our experiments have demonstrated that the robot helicopter can hover and follow a path. We can configure the sensor field with arbitrary paths and change them dynamically. The helicopter can interact with the sensor field and retrieve path segment information. A second joint field experiment, targeted for June 2003, will be used to collect navigation data.

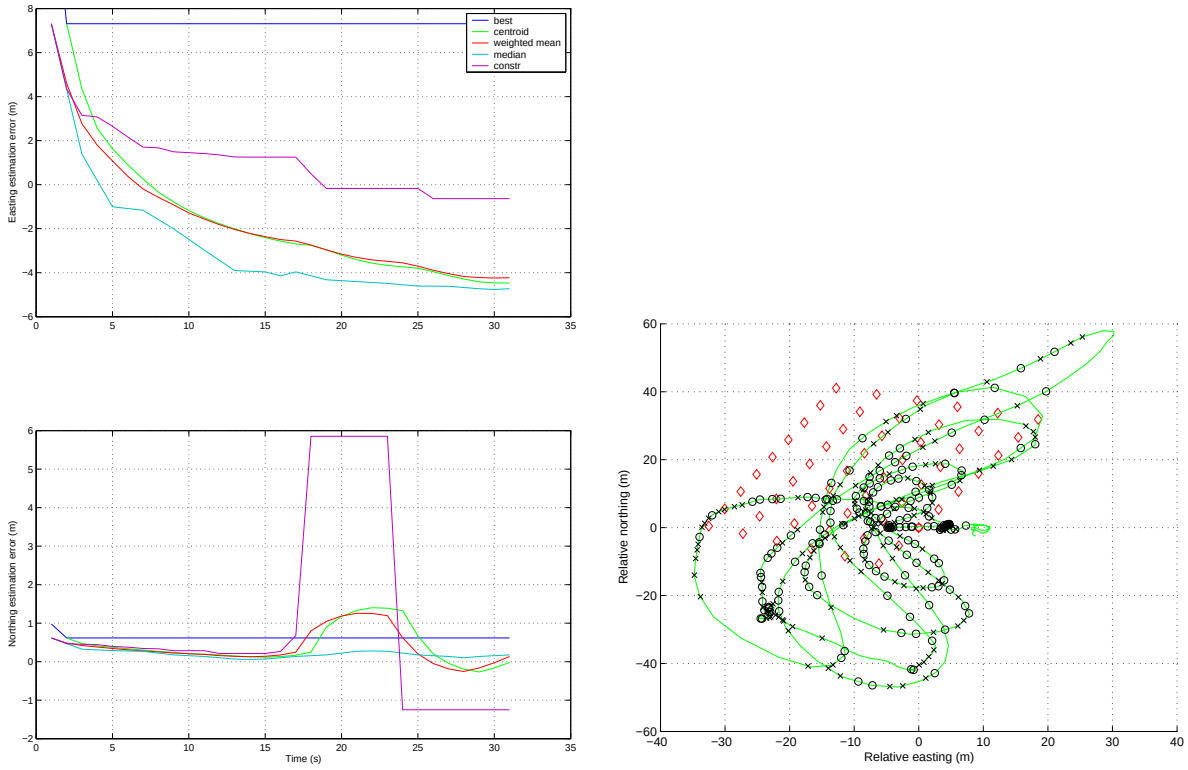


Figure 2: (Left and Center) Results of offline localization by GPS, evolution of different estimates with time for our 5 localization methods. (Right) Localization of a Mote using the helicopter. The helicopter path is shown with GPS reception marked: **o** denotes a good packet and **x** a bad packet.

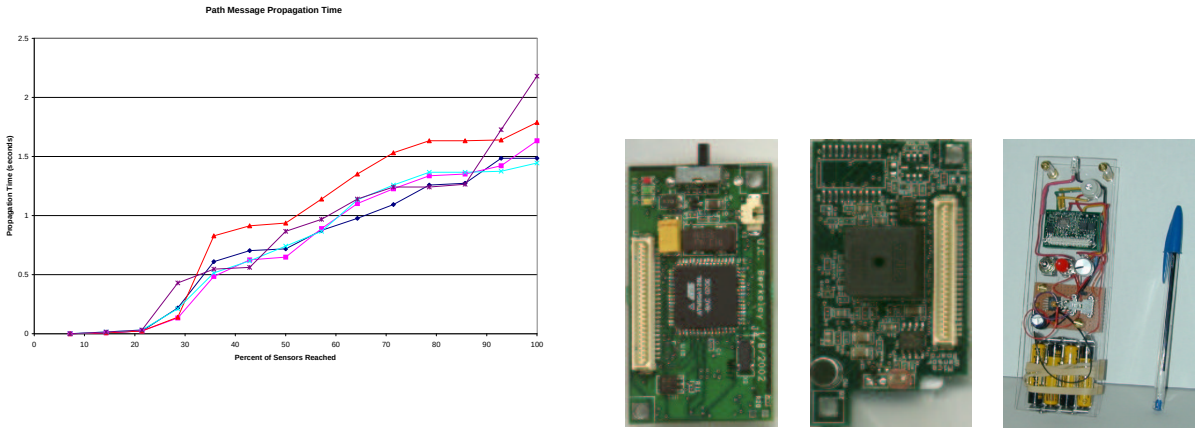


Figure 3: (Left) Path propagation time for 5 different paths over a grid of 54 Mote sensors. The $y$ axis shows the time and the $x$ axis the percentage of the sensors that are on the path and have seen the path message. (Center) A Mote board and Mote sensor board. (Right) The Flashlight device.

### 3.4.1 Lessons Learned

We have gained several insights into networked robots. Data loss is not rare in sensor networks. This is due to network congestion, transmission interference, and garbled messages. We observed that the transmission range of 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 the nodes in proximity of each other try to send packets at the same time. For a sensor network, because of its small memory and simplified protocol stack, congestion is a big 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.

### 3.4.2 Extension to Guiding Humans

The ideas behind guiding the navigation of robots can be extended to guiding humans augmented by a hand-held device. It is based on the metaphor of a flashlight and is called a *sensory Flashlight*, see Figure 3(Right). It comprises an analog compass, alert LED, pager vibrator, and a Berkeley Mote [11].

When pointed in a specific direction, the Flashlight collects information from all the sensors located in that direction and provides its user with feedback. For the navigation task this feedback consists of a vibration when the flashlight is pointed in the direction of travel computed by the network. The device can also issue commands to the sensors in the direction it is pointing.

We have deployed 12 Mote sensors along corridors in our building and used the Flashlight and the path guidance approach presented here to guide a human user out of the building. Figure 3(Right) shows the map. The Flashlight interacted with sensors to compute the next direction of movement toward the exit. For each interaction, the user did a rotation scan until the Flashlight was pointed in the direction computed from the sensor data. The user then walked in that direction to the next sensor. Each time we recorded the correct direction and the direction detected by the Flashlight. The directional error was 8% (or 30 degrees) on average. However, because the corridors and office doorways are wide, and the sensors sufficiently dense, the exit was identified successfully. The user was never directed toward a blocked or wrong configuration. An interesting question is how dense should the sensors be, given the feedback accuracy.

# 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. 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 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 the rest of the Dartmouth sensor network team: Qun Li and Michael de Rosa. Support for this work was provided through the NSF awards 0225446, EIA-9901589, IIS-9818299, and IIS-99R12193, ONR award N00014-01-1-0675 and the Darpa TASK program. We are grateful for it.

# References


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

[2] Distributed localization in wireless sensor networks, Available at citeseer.nj.nec.com/464015.html, 2002.

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

[4] P. Bergamo and G. Mazzimi. Localization in sensor networks with fading and mobility. In *Proceedings of IEEE PIMRC*, Lisbon, Portugal, 2002.

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

[6] G. Buskey, J. Roberts, P. Corke, P. Ridley, and G. Weyth. Sensing and control for a small-size helicopter. In *Experimental Robotics VIII*, pages 476–486, Ischia, Italy, 2002.

[7] 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.

[8] Deborah Estrin, Ramesh Govindan, and John Heidemann. Embedding the internet. *Communications of ACM*, 43(5):39–41, May 2000.

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

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

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

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

[13] J.-C Latombe. *Robot Motion Planning*. Kluwer, New York, 1992.

[14] Qun Li, Michael de Rosa, and Daniela Rus. Distributed algorithms for guiding navigation across a sensor net. In *Dartmouth Computer Science Technical Report TR2002-435. Submitted to MobiCom 2003*, 2003.

[15] 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.

[16] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[17] Elizabeth Royer and C-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. In *IEEE Personal Communication*, volume 6, pages 46 – 55, April 1999.