12-1-2004

STAM: A System of Tracking and Mapping in Real Environments

Ying Zhang

Palo Alto Research Center

Lee Ackerson

Palo Alto Research Center

David Duff

Palo Alto Research Center

Craig Eldershaw

Palo Alto Research Center

Mark Yim

University of Pennsylvania, yim@grasp.cis.upenn.edu

Follow this and additional works at: https://repository.upenn.edu/meam_papers

Recommended Citation

Zhang, Ying; Ackerson, Lee; Duff, David; Eldershaw, Craig; and Yim, Mark, "STAM: A System of Tracking and Mapping in Real Environments" (2004). *Departmental Papers (MEAM)*. 37.

https://repository.upenn.edu/meam_papers/37

Copyright 2004 IEEE. Reprinted from *IEEE Wireless Communications*, Volume 11, Issue 6, December 2004, pages 87-96.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/meam_papers/37

For more information, please contact repository@pobox.upenn.edu.

STAM: A System of Tracking and Mapping in Real Environments

Abstract

We have implemented a system of tracking mobile robots and mapping an unstructured environment, using up to 25 wireless sensor nodes in an indoor setting. These sensor nodes form an ad hoc network of beacons, self-localize with respect to three anchor nodes, and then track the locations of mobile robots in the field. The system described here was motivated by search and rescue applications, and has been demonstrated in real physical environments.

Disciplines

Applied Mechanics | Engineering | Mechanical Engineering

Comments

Copyright 2004 IEEE. Reprinted from *IEEE Wireless Communications*, Volume 11, Issue 6, December 2004, pages 87-96.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

STAM: A SYSTEM OF TRACKING AND MAPPING IN REAL ENVIRONMENTS

YING ZHANG, LEE ACKERSON, DAVID DUFF, AND CRAIG ELDERSHAW,
PALO ALTO RESEARCH CENTER
MARK YIM, UNIVERSITY OF PENNSYLVANIA

The authors have implemented a system of tracking mobile robots and mapping an unstructured environment, using up to twenty-five wireless sensor nodes in an indoor setting.

ABSTRACT

We have implemented a system of tracking mobile robots and mapping an unstructured environment, using up to 25 wireless sensor nodes in an indoor setting. These sensor nodes form an ad hoc network of beacons, self-localize with respect to three anchor nodes, and then track the locations of mobile robots in the field. The system described here was motivated by search and rescue applications, and has been demonstrated in real physical environments.

MOTIVATION

Over the last half century, practical applications of robotics have been a topic of much research. A major hurdle to using robots in the real world is the unstructured and unpredictable nature of the environment.

Urban search and rescue (USAR) environments, where there may be partially collapsed buildings from bombings or earthquakes, are a particular example identified as a challenge for robotic research [1]. Even though the current technology is still too limited to enable fully autonomous systems for USAR, tele-operated robots can aid in finding victims. More generally, these techniques can be employed to explore unknown and hazardous environments such as deep sea, lunar and planet surface operations, or military operations in urban terrain.

A critical aspect of controlling a tele-operated robot in unknown and unstructured environments is the operator's awareness of the robot's position. With just a video feed from the robot, it has been reported that operators frequently become confused as to the robot's orientation and location. This is especially prevalent in a dynamic unstructured environment that may be partially occluded by smoke and dust.

To be truly useful in USAR, a key component of the system is reporting to rescue personnel the location of the victim. This requires the robot to know where it is at all times and the ability to generate a human readable map of the environment as the robot explores.

Using a robot to create a map of the environ-

ment while tracking its location is not a new concept. Much research has gone into simultaneous localization and mapping (SLAM) over the last decade (e.g., [2]). SLAM typically uses ranging sensors (e.g., laser radar) that return an array of relative distances to features in the environment. These distance measurements, in conjunction with a model of the robot's motion, allow a SLAM algorithm to construct a map showing the location of those features and of the robot as the robot moves.

There are several problems with implementing current SLAM techniques in the USAR scenario. First, accurate sensors such as laser radar are normally big and heavy: carrying such a device may limit a robot's mobility in small or confined spaces. Second, creating accurate motion models for robots moving through unstructured dynamic environments (e.g., with shifting rubble and dirt using snakelike robots) is a hard problem in itself. Third, while SLAM in structured environments such as inside buildings or city streets has been shown in the literature, unstructured non-flat terrain is still an open problem.

Ad hoc wireless sensor networks [3] have been an active research topic for several years. Sensor networks have been applied to environmental monitoring, traffic control, building management, and object tracking. In many of these applications, localization has been a key component, where Global Positioning System (GPS)-based solutions are either too expensive or not applicable (e.g., indoor environments). Various localization algorithms for sensor networks have been developed (e.g., [4–6]); however, most work has been done in simulation, and little applied to real systems.

We have implemented a complete system for tracking and mapping (STAM) and demonstrated it using 25 sensor nodes, three of which are anchor nodes mounted in a triangle frame, and two of which are mounted on two of the mobile robots. The sensor nodes are deployed into an unknown environment, with the anchor frame placed at the entrance. A deployment mechanism has been implemented but is out of the scope of this article. The sensor nodes form an

The ultrasound is used in conjunction with the radio on the Mica2 to detect distances between sensor nodes by measuring time of flight for the ultrasonic pulses.

ad hoc network of beacons and localize themselves with respect to the anchor nodes using the pairwise ranging data. The localized sensor nodes are then used to track the locations of the mobile robots in the field.

This method can be used for both tele-operated and fully autonomous robots. For the initial practical experimentation, tele-operated snake-like robots are used. A human operator at a remote station controls the motion of the robot to explore the environment via cameras mounted on the robot. The operator maintains better environmental awareness by tracking the robot and viewing the global location of the robot in the generated map.

The rest of the article is organized as follows. We describe the overall system architecture, including both hardware and software components and the user interface for mapping and tracking. We present the wireless communication protocol for performing ultrasound ranging and data collection. We present the distance model of the ultrasound ranging data and the filtering process for localization. We focus on three recently developed algorithms and the automatic procedure of combining results from different algorithms. We then show some experimental results of the system in indoor environments.

STAM OVERALL SYSTEM ARCHITECTURE

STAM is composed of three parts: the sensor nodes, the remote station, and the mobile robots.

Each sensor node is a Berkeley mote (Mica2) augmented with an ultrasonic ranging device developed at the Palo Alto Research Center (PARC). The Berkeley mote platform [7] is a widely used platform for sensor networks. Each Mica2 has an Atmel microprocessor (120 kb flash and 4 kb RAM) with a 433 MHz radio transceiver. Optional sensor boards for the Berkeley motes can hold various sensors, including light and temperature sensors, microphones, and accelerometers. We use this same interface to mount the ultrasound board. Sensor nodes are also labeled with identification numbers and augmented with LEDs that can be lit to serve as visible beacons in the dark to lead rescue personnel to the victims in USAR.

The ultrasound is used in conjunction with the radio on the Mica2 to detect distances between sensor nodes by measuring time of flight for the ultrasonic pulses. Figure 1a shows the electronics of a sensor node. The ultrasonic receiver and transmitter can be seen on top of the Mica2 (two metal cans). The bare ultrasound devices produce a fairly tight beam transmission. As can be seen in Fig. 1a, a plastic hemisphere was attached above the transmitter to convert the beam to a nearly flat spherical output.

Similarly, a cone is mounted on top of the receiver to collect signals from a wide range of input angles. This device can accurately measure distances of up to 15 ft, which are suitable for indoor-room-size environments.

The mote with the ultrasound device is housed inside a labeled translucent white plastic ovoid shell (Fig. 1b) that provides snag-free structural protection from the environment; the

surface of the shell glows when lit from inside.

The weight distribution within the shell is such that these sensor nodes are self-righting after deployment; approximate vertical alignment of all the ultrasound transmitters and receivers is important.

Once sensor nodes have been deployed, the localization process can be started by getting a distance matrix of the network using the ranging devices. After localization, the sensor network serves two purposes:

- To track mobile robots in the field to help the robot driver navigate
- To act as an active landmark (e.g., selectively turning on a light) in the environment to guide rescuers to victims in USAR

The Berkeley motes are resource constrained in terms of memory and computation power. Given the large data sets and the requirement to process them in real time, localization computation is performed at a PC. This design does not scale to hundreds of thousands of nodes; however, for indoor search and rescue applications, the maximum number of nodes within a room is limited, so centralized computation does not become the bottleneck in this case. The STAM host environment is written in Matlab and can incorporate any localization algorithm written in Matlab by adding an interface to Berkeley's Calamari application programming interface (API) [8]. So far we have integrated more than seven localization algorithms and have done extensive testing of them (including some developed at Berkeley, Stanford, and PARC). The localization results can be compared in terms of either *simulation errors* or *data errors*. The simulation error is defined to be the average of the differences between the actual and estimated locations. The data error is defined to be the average of the differences between the actual distance obtained from range data and the distance calculated from the estimated locations. These two types of errors are related. However, simulation errors are meaningful only for simulation, while data errors can be calculated for both simulated and real localizations. The three best algorithms were selected and used for the real deployment experiments.

The host environment also provides a user interface for mapping and tracking, and can be run in both simulation and deployment modes. The simulation mode is used for developing and debugging localization algorithms. Figure 2 shows a snapshot of the graphical user interface (GUI) for STAM. The center of the GUI is the draw/display panel for the distance graph of sensor nodes. In simulation mode, one can draw an arbitrary distance graph with any predefined noise model and select any set of the nodes to be anchor nodes. One can also load from a file any graph with generated or real experimental data. The estimated locations will be displayed together with the ground truth after localization. In deployment mode, this panel is used to place anchor and/or sensor nodes, to display the connectivity between sensor nodes, and to show the positions of sensor nodes and mobile robots after localization.

On the left side, there are choices for different display modes, buttons for anchor selections,

(a)

(b)

Figure 1. A Berkeley mote with ultrasonic ranging device: a) exposed; b) within its self-righting shell.

Figure 2. A snapshot of the STAM graphical user interface.

For the real deployment, a serial connection from the host to the Mica2 base station is established to pass commands from the host to the sensor network and pass data from the sensor network to the host.

■ **Figure 3.** A snake robot for search and rescue applications.

save/load files of generated or actual data sets, and a textbox for specifying data noise functions. One can choose different localization algorithms, or do the “best” localization using the algorithms in the system. Both the simulation errors and data errors are shown after localization.

The right side of the GUI is mainly for deployment. There are buttons for controlling the state of each individual sensor node and tracking the positions of the mobile sensor nodes mounted on the mobile robots. For search and rescue applications, the remote tele-operator can also add victim conditions and landmarks from the environment (using cameras and other types of sensors mounted on the mobile robots) to this map to give more information to rescue personnel.

For real deployment, a serial connection from the host to the Mica2 base station is established to pass commands from the host to the sensor network and pass data from the sensor network to the host. Three anchor nodes mounted on a triangle frame are placed on the ground before sensor deployment. There are two types of deployment:

- Deploy and localize one sensor node at a time.
- Deploy and localize all the sensor nodes at once.

In the former case, the operator clicks on the display panel to deploy a new node, and global localization is recomputed after each deployment using the new range data. In the latter case, the operator hits the LocalizeAll button on the right panel to start the localization process. After localization, one can track any one of the mobile robots in the field. The traces of the robots (showing the history of their positions) are displayed on the map as well.

There are no restrictions on the type of mobile robots for STAM; one can have any type of mobile robot that is flexible to traverse through small spaces, robust enough to walk over uneven terrains, and able to carry the weight of various sensing devices. The experiments performed here use snakelike robots developed from the modular robot PolyBot [9]. Figure 3 shows a picture of one of our robots, Hansel. A snakelike shape is well suited to moving through unstructured environments: its small cross-section allows the body to move through narrow gaps, yet its long reach allows it to cross over larger obstacles. The front of the robot has a video camera, as well as sensors (microphone,

CO₂ and temperature sensors) to help locate and diagnose victims. On the middle of the snake a sensor node is mounted. The position of this sensor node can be tracked by the sensor network via localization. Hansel was used for the search and rescue competition organized by the National Institute of Standards and Technology (NIST) in summer 2004 and won third place.

COMMUNICATION PROTOCOL FOR RANGING

From the software system point of view, STAM has two parts: the embedded portion is implemented on Mica2 in NesC/TinyOS, and the host portion is implemented on a PC in Matlab.

In general, there are two ways to measure distances between two points using ultrasound: signal strength or time of flight (TOF). We use TOF because it is more robust in the presence of noise and does not need a signal fading model. The distance is measured using the simultaneous transmission of radio and ultrasound pulses. A 16-bit timer in Mica2 is used to measure the time difference between the receipt of the radio and ultrasound pulses: the timer is reset to zero when receiving a radio signal indicating the start of an ultrasound transmission and stopped by an interrupt triggered by the arrival of the ultrasound wave. The radio transmission is effectively instantaneous, so this time difference and the speed of sound through air give a good approximation of the distance between the transmitter and receiver. The clock is set to be 1 MHz, which gives a high precision measurement: each tick corresponds to about 0.01 in. With 16 bits the counter will overflow in about 0.07 s, limiting the maximum detectable distance to about 50 ft. However, the reliable distances measured by this device are only up to about 15 ft. For larger distances, the measurement is particularly subject to inaccuracy due to reflections.

A Mica2 base station is connected to the remote station via a serial cable, which acts as a gateway between the sensor network and the host. The Mica2 radio transmission range extends beyond 30 ft, so single-hop messaging is sufficient for localization in an area the size of a room. This is adequate for our immediate application; for longer-range localization, the method can easily be extended to utilize a multihop network.

There are two types of packets: *range data* packets and *command* packets. Range data packets play double roles:

- Broadcasting a radio signal for the start of the ultrasound transmission from this node
- At the same time transmitting the current set of range data, distances from the neighboring nodes to this node, to the host

Command packets are used to change the states and parameters of the sensor nodes. Range data packets are broadcast from the sensor nodes. Command packets are sent from the host to either a specified sensor node or all sensor nodes in the field.

Each sensor node is in one of four states: OFF, IDLE, TRANSMITTING, and RECEIVING. All the sensor nodes are in the OFF state when the power is switched on. In the OFF state, a sensor node does not respond to any range data packet, and no ranging measurement is taken. This state is used when a sensor node is turned on but has not yet been deployed to its location. In the IDLE state, a node will respond to a radio signal indicating the start of the ultrasound transmission from the node that sent the radio signal, and move into the RECEIVING state. Upon entering the RECEIVING state, the 16-bit 1 MHz timer is set to zero. While in this state an interrupt will arrive triggered by either the receipt of an ultrasound wave or the overflow of the timer (after about 0.07 s) if the sound is blocked by obstacles. In the former case, the value in the timer is processed and recorded as a measurement of the distance between the sound source and this node. In either case, the sensor node moves back to the IDLE state after the interrupt. While in the IDLE state, a node can also go to the TRANSMITTING state, either commanded by the host, triggered by a timer, or responding to receipt of range data from a new or moving sensor node. Upon entering the TRANSMITTING state, the node broadcasts a range data packet, transmits an ultrasound pulse for a period of time (e.g., 50 ms), and then moves back to the IDLE state.

The command packets sent from the host have two bytes of data: one byte for command type and one byte for command value. The commands include START (moving from the OFF state to the IDLE state), STOP (moving from the IDLE state to the OFF state), TRANSMIT (transmitting the ultrasound for a specified number of times), and RESET (clearing the range data array). Commands are also used for setting parameters of the sensor node from the host, such as the period between two ultrasound transmissions or the flag indicating whether or not to respond to the receipt of range data from a new or moving sensor node.

The range data packets sent from the sensor nodes consist of the sender's address and an array of up to six sound-source/value pairs.

To obtain the current range data to transmit, we keep an array of all the sound-source/value pairs with priorities, where higher priorities indicate more recent data. Each incoming ultrasound pulse detected generates a new sound-source/value pair, which is recorded with the highest priority. The priority of a stored entry is decreased each time after this entry is transmitted. The six pairs with the highest priorities are the ones packed into the range data packet. If the sensor node is new to the system

or if it is moving, the number of sound-source/value pairs is zero.

One important design issue for this type of ranging system is to guarantee that no two nodes transmit at the same time. One simple strategy is to use round-robin among the sensor nodes so that each sensor node gets a time slot to transmit. However, such a strategy is not very efficient when deploying nodes one at a time, or when most of nodes are stationary while only a couple of nodes are moving. In these cases, the new node or moving nodes should receive more time slots, and those who have recently obtained the range data from the new or moving sensor nodes should transmit the data to the host.

In this implementation, each sensor node has an ID. If the sensor nodes are deployed one by one, the nodes are deployed in the order of increasing IDs. The moving nodes always have the highest IDs. After receiving a range data packet from a new or moving sensor node (whose number of sound-source/value pairs is zero), if the source ID is greater than its own ID and the response flag is set, the node will transmit after $k(ID_s - ID)$ s where k is a constant large enough so that there are no two nodes whose transmit period could overlap in time. In this implementation, k is set to be twice the minimum period between two transmissions. This parameter and the response flag can be set from the host.

The user can also choose to localize after all nodes have been deployed. In this case, each node takes turns to transmit a couple of times in a round-robin fashion. No response flag should be set. The TRANSMIT commands are sent from the host with a fixed sampling time, which guarantees that no two nodes transmit at the same time.

When the system enters the tracking mode (i.e., calculating the position of a moving sensor node), a RESET command is sent to the moving node to clear the old range data array, and the response flag is set for all the nodes in the field.

The host receives all the range data packets through the Mica2 base station. A distance matrix is built up, and is used in localization for tracking and mapping.

DATA FILTERING FOR LOCALIZATION

The raw range data collected from the ultrasound device described in the previous section are noisy. It is necessary to implement mechanisms that deal with noisy data on both the embedded and host sides.

At the embedded side, we have implemented an efficient moving average. It is observed that when the distances are large, there are increasing chances of data errors due to reflections of nearby objects. For most localization situations, bad data are worse than no data, so all large values are discarded. For every sound source, the minimum and maximum values, min and max , are recorded. When a new piece of data comes in, if the value v is larger than the predefined maximum value limit, the value is discarded. Otherwise, the current value is obtained as follows: if $v > max$, $max \leftarrow v$; otherwise, $max \leftarrow (max + v)/2$; if $v < min$, $min \leftarrow v$; otherwise,

At the embedded side, we have implemented an efficient moving average. It is observed that when the distances are large, there are increasing chances of data errors due to reflections of nearby objects.

■ Figure 4. The ultrasound time of flight distance model.

$\min \leftarrow (\min + v)/2$. The current value is then calculated as $(\min + \max)/2$.

To accurately model the relationship between real distance and the TOF obtained from this ultrasound device, tests were done between two nodes (in an open area) at a variety of known distances. For each distance, 20 measurements were taken. Figure 4 shows the average data and error bounds (minimum and maximum) with respect to distances between two nodes. The graph shows that the ranging error is small and independent of distance when there are no reflections from the environment.

From these data, we obtained the distance model as $d \leftarrow (v - 1200)/70$ where v is the range data value and d is the number of inches between two nodes. Figure 4 shows that the model fits the data very well over a wide range of distances (1–15 ft).

Even though ultrasound TOF distance measurements are quite accurate in an open field, for indoor environments most errors are due to reflections from walls, ceilings, and objects nearby. On the host side, two other data filtering techniques are used, *symmetric distance enforcement* and *triangular inequality enforcement*, both utilizing global information from the distance matrix.

Symmetric distance enforcement: In most cases, distance obtained from node A to node B , $d(A,B)$, is not the same as that from node B to node A , $d(B,A)$. In fact, one of these measurements may not even exist. On the assumption that the larger measurement is the result of an indirect path (one involving a reflection), $\min(d(A,B), d(B,A))$ is taken to be the value for the distance between A and B . If only one measurement exists (assuming the other value to be infinite), the value of this measurement is used.

Triangular inequality enforcement: Given any two points A and B in space, we should have $d(A,B) \leq d(A,C) + d(C,B)$ for any C if both $d(A,C)$ and $d(C,B)$ exist. If such a relationship

does not hold, we choose to ignore the connection between A and B since bad data are worse than no data for most situations.

Both of these techniques have been shown to be very useful for filtering out bad data. However, triangular inequality enforcement will reduce connectivity of the network if the range data are noisy due to obstacles and reflections. For localization algorithms that are sensitive to connectivity, the use of this preprocessing may obtain worse results. In practice, both techniques have improved the results of localization significantly in indoor environments.

LOCALIZATION ALGORITHMS

The generalized localization problem can be formalized as follows: given a set of n nodes and a set of range data d_{ij} (distances between nodes i and j), find a set of location assignments x_k , $k = 1 \dots n$ such that $|x_i - x_j| = d_{ij}$. Since the range data may be noisy and/or redundant, this problem can be formalized as a least squares problem, that is, minimizing $\sum (|x_i - x_j| - d_{ij})^2$. To obtain absolute locations, at least three anchor nodes (nodes with known positions) are needed for a 2D localization. For a map of relative positions, no anchor nodes are theoretically needed, although most localization algorithms require them anyway.

LOCALIZATION INTERFACE

For the search and rescue application of mapping only a single room, 15–20 sensor nodes will be deployed at a time, so a centralized algorithm suffices. The centralized algorithms written in Matlab run on the host that is connected to the Mica2 base station via a serial connection. The host receives all the range data sent out by the sensor nodes and forwards it to the STAM localization interface which builds the connectivity and the distance graph. The STAM localization interface adheres to the Berkeley Calamari API [8]. A data set for localization of N nodes is a structure including:

- **xy:** $N \times 2$ array for actual positions of the N nodes
- **distanceMatrix:** $N \times N$ matrix for the actual pairwise distances between any two nodes
- **connectivityMatrix:** $N \times N$ matrix whose element is 1 if two nodes are connected and 0 otherwise
- **kd:** $N \times N$ matrix whose elements are distance values obtained by ranging if one node can hear from another, and -1 otherwise (note that **kd** may not be symmetric)
- **nodeIDs:** an $N \times 1$ array of node IDs
- **anchorNodes:** an $M \times 1$ array of node IDs of M anchor nodes
- **xyEstimates:** $N \times 2$ array for estimated positions of the N nodes

In this structure, **connectivityMatrix**, **kd**, **nodeIDs**, and **anchorNodes** are the input, and **xyEstimates** is the output of a localization algorithm (although **connectivityMatrix** can be generated from **kd**); **xy** and **distanceMatrix** are ground truth and are only used for simulations.

More than seven localization algorithms written by various groups (Berkeley, Stanford,

and PARC) have been integrated into this environment and evaluated interactively through the user interface of STAM. A survey of different localization algorithms and their performance evaluations is beyond the scope of this article. Here, we only briefly introduce three of the most successful localization algorithms tested in this environment so far, and compare their pros and cons. Then we present a control flow that combines these three algorithms, and obtains robust tracking and mapping solutions.

LOCALIZATION ALGORITHMS

In all these algorithms, we assume that there are at least three anchor nodes whose positions are known, and that the input data set has been pre-processed using the techniques described in the previous section.

MDS-based localization: The heart of MDS-based localization is multidimensional scaling (MDS) [5], a data analysis technique that transforms proximity information into a geometric embedding. The kernel of the MDS-based localization algorithm MDS-MAP(C) consists of three steps:

- Compute the shortest *path distance* between all pairs of nodes. A distance matrix is obtained where each entry is the shortest estimated distance (path distance) between two points. The distance matrix is used to approximate the Euclidean distance.
- Apply MDS to the distance matrix, retaining the first two (or three) largest eigenvalues and eigenvectors to construct a 2D (or 3D) relative map.
- Transform the relative map to an absolute map using the known positions of anchor nodes.

For uniformly deployed sensor nodes (e.g., grid deployment), this method works very well. For irregular sensor placement, the distance matrix no longer approximates the Euclidean distance well. MDS-MAP(P) solves this problem by building local maps and patches them to form a global map. MDS-MAP(P) has been shown to work very well for irregular types of deployment. For both MDS-MAP(C) and MDS-MAP(P), a refinement step can be added. The refinement step uses the position estimates of nodes in the MDS solution as an initial solution, then applies least squares minimization to improve the match between the measured distance and the distances in the solution.

The advantage of the MDS-based localization is that it works well with few anchor nodes (or in the extreme case without anchor nodes). The disadvantages are that it does not work well for irregular networks with concave connectivity (even with MDS-MAP(P)); the refinement step is generally slow; and the solutions are subject to local minima.

SDP-based localization: Semidefinite Programming (SDP) is a formal mathematical approach to solving this problem [4]. In SDP-based localization, the minimization problem is transformed to an SDP; a unique and also computationally efficient solution can be obtained via well-developed SDP solvers. It can also be shown that if the problem is *localizable* (i.e., a

unique solution exists), it is in fact the least squares solution. This means that the solutions of the SDP-based method are generally of high quality.

A side-benefit of SDP-based methods is that as part of the algorithm, estimates are developed as to the quality of the point estimation. This allows easy detection of erroneous sensors (which can then be disabled). However, SDP-based methods do require at least three anchor nodes in the field. Also, it does not work well on data sets with large amounts of data noise. A distributed version of the SPD-based method does exist, but its computational requirements are beyond that of resource-limited embedded devices such as Berkeley motes.

Incremental localization: Incremental localization [6] uses an incremental least squares algorithm (ILS). The ILS-based iterative localization is fast, anytime, and scalable. It uses an error registry mechanism to choose those neighboring nodes whose location is best known (in terms of noise and error propagation model), and estimates its own location based on those selected nodes. In this way, updates on the positions of nodes are propagated outward from the set of anchor nodes (whose locations were known initially).

The ILS algorithm can be distributed and requires only limited computation on each sensor node. However, it places greater requirements on the positions of anchor nodes (at least three in a connected region) and connectivity of the sensor network (at least three known locations in the neighborhood at a time) to propagate the location estimation. Furthermore, this method is sensitive to errors if the error model is not accurate.

LOCALIZATION PROCESS

We have tested all the algorithms integrated in STAM on various distance graphs generated by simulation or obtained from real data. MDS, SDP, and ILS are the best algorithms overall, and each has advantages in certain situations. Figure 5 shows the three algorithms applied to three test cases, all with identical node distributions. In the first case, there is one node with low connectivity (connected to only two nodes); in the second case, all nodes have at least three connections and no data noise. Both use “perfect” data sets (i.e., no noise occurs in the measured distance). In the last case, the connectivity is identical to the second case but with noise added. We can see that MDS has almost identical performance for all three cases. SDP is robust to low connectivity but sensitive to noise, ILS is sensitive to both low connectivity and noise. For the third case, because of the use of triangular inequality enforcement, the connectivity is reduced, causing the total failure of ILS. In these cases, MDS works most robustly, but both SDP and ILS work better than MDS in low data noise and high connectivity.

For a real situation, it may not be known a priori which localization algorithm works best. We devised a meta-strategy that runs these algorithms in turn and selects the best result

The ILS algorithm can be distributed, and requires only limited computation on each sensor node. However, it places greater requirements on the positions of anchor nodes and the connectivity of the sensor network to propagate the location estimation.

Figure 5. Localization results of three algorithms for three cases: I (there is a node with low connectivity), II (good connectivity and no data noise), and III (same as II with large data noise). The three anchor nodes are positioned in a triangle on the left bottom. The estimated locations are shown with a box connected to the actual locations. SE indicates simulation error and DE indicates data error.

obtained. Since most localization algorithms will not work well if there are nodes with low connectivity, the connectivity is checked before applying the localization algorithms. We call a node *well connected* if it has at least *three* connections to other nodes. A network is *well connected*, *connectedAll*, if all nodes are well connected. For ILS, well connectivity is insufficient; sequential well connectivity is a stronger condition. Given an order of nodes, a node is *sequentially well connected* if it connects to at least *three* nodes in the lower order of the sequence. A network is *sequentially well connected*, *sconnectedAll*, if all nodes are sequentially well connected. Two localization results can be compared based on their data errors: the differences between the actual distance obtained from the range data and the distance calculated from the estimated locations after localization. The better solution is taken to be the one with the smaller data error. The pseudo code of this strategy is follows:

```

IF connectedAll(dataSet) DO
    dataSet1 = MDSSolver(dataSet);
    dataSet2 = SDPSolver(dataSet);
    best = compareTwo(dataSet1, dataSet2);
    IF sconnectedAll(dataSet) DO
        dataSet3 = ILSSolver(dataSet);
        best = compareTwo(best, dataSet3);
    END
END

```

The meta localization strategy will select the solution with the smallest data error among results from MDS, SDP, and, if the network is sequentially well connected, ILS.

We have created robust control flows to localize all nodes at once, localize one new node at a time, and localize a moving node.

For localizing all nodes at once, there is a loop where first all nodes transmit in turn and then the localization procedure is applied. If there are nodes that are not well connected, those nodes will transmit a couple of more times,

Figure 6. An example of experimental test: a) a disaster field with sensor nodes; b) the result of mapping and tracking.

and the process is repeated. If, after a certain number of trials, there still remain nodes that are not well connected, those nodes will be removed from the distance graph. The process continues until either a minimum number of trials are finished successfully or a maximum number of trials has been reached. From these successful trials, the best localization result will be used as the final result. For localization one node at a time, the new node will transmit a couple of times, and those nodes who hear the ultrasound transmission will reply with range data. Then localization will be attempted. As with the previous case, this process will be repeated a couple of times. For tracking a moving node, all the stationary nodes are set to be anchor nodes. The tracking process is similar to localizing one new node at a time, except that each time a new data set is generated from the stationary distance graph and the new connections between the moving node and stationary nodes.

EXPERIMENTAL RESULTS IN REAL ENVIRONMENTS

We have tested this system in real physical environments. In these experiments, we have three anchor nodes mounted on a triangular frame. The sensor nodes are randomly placed on the floor with pipes, stones, and other types of debris simulating a disaster scene. The anchor frame is placed on the side of the field, since in real search and rescue cases it would be placed near the entrance.

Figure 6a shows an example of such a field, and Fig. 6b shows the localization result, where circles are estimated locations, stars denote the anchor nodes, dots mark the path of one of the mobile robots, and a cross indicates the current position of the robot. For successful trials, the data errors are ranging from 4 to 10 in a 300 in $\times$ 300 in field, depending on topologies and noises. Large errors imply too many reflections or too few connections. Results of large data errors are automatically filtered out.

CONCLUSION AND FUTURE WORK

There has been a lot of work recently on localization in sensor networks (e.g., [10–15]), but not many real systems have been built so far. Motivated by search and rescue applications, we have designed and developed a complete system for mapping an unknown field using sensor networks and tracking mobile robots in that field. The generated map not only provides guidance during tele-operation of the mobile robots, but also a tool for navigation by rescuers at the scene after the robotic exploration.

We have integrated a variety of localization algorithms into the system and studied their performance using both simulated and real data. We have developed a robust ranging mechanism, data filtering strategies, and control flows for getting reliable localization results in real environments.

In the future we will use STAM to improve and integrate more localization algorithms, study noise properties and network topologies of various environments, investigate incremental localization algorithms for tracking, and design and implement distributed and scalable localization solutions.

ACKNOWLEDGMENTS

The authors wish to thank Dan Lerner for his input and assistance. Thanks also to Feng Zhao and Markus Fromherz for their support and comments. This work is funded in part by Defense Advanced Research Project Agency (DARPA) contract # F33615-01-C-1904.

REFERENCES

- [1] "Sessions on Robots and Systems for Disaster Response (i) and (ii)," *Int'l. Conf. Intelligent Robots and Sys.*, 2003.
- [2] S. Thrun, W. Burgard, and D. Fox, "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots," *Machine Learning and Autonomous Robots* (joint issue), 1997.
- [3] D. Culler and W. Hong, "Wireless Sensor Networks," *ACM Commun.*, vol. 47, no. 6, 2004.
- [4] P. Biswas and Y. Ye, "Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization," *ACM/IEEE Conf. Info. Proc. in Sensor Networks*, 2004.

Motivated by search and rescue applications, we have designed and developed a complete system for mapping an unknown field using sensor networks, and tracking mobile robots in that field.

In the future we will use STAM to improve and integrate more localization algorithms, study noise properties and network topologies of various environments, and investigate incremental localization algorithms for tracking.

- [5] Y. Shang et al., "Localization from Connectivity in Sensor Networks," *IEEE Trans. Parallel and Distrib. Sys.*, vol. 15, Oct. 2004.
- [6] F. Zhao et al., "Incremental Node Localization with Error Propagation Control in Ad Hoc Networks," Tech. rep. P200310265, PARC, 2004.
- [7] Berkeley wireless embedded systems, 2003; <http://webs.cs.berkeley.edu/>
- [8] K. Whitehouse, "Calamari: A Localization System for Sensor Networks," 2003; <http://www.cs.berkeley.edu/~kamin/calamari/>
- [9] M. Yim, Y. Zhang, and D. Duff, "Modular Robots," *IEEE Spectrum*, Feb. 2002.
- [10] L. Doherty, L. E. Ghaoui, and K. Pister, "Convex Position Estimation in Wireless Sensor Networks," *Proc. INFOCOM 2001*, Anchorage, AK, Apr. 2001.
- [11] J. Hightower and G. Boriello, "Location Systems for Ubiquitous Computing," *IEEE Comp.*, vol. 34, Aug. 2001, pp. 57–66.
- [12] D. Niculescu and B. Nath, "Ad Hoc Positioning System," *IEEE GLOBECOM*, Nov. 2001.
- [13] C. Savarese, J. Rabaey, and K. Langendoen, "Robust Positioning Algorithm for Distributed Ad-hoc Wireless Sensor Networks," *USENIX Tech. Annual Conf.*, Monterey, CA, June 2002.
- [14] A. Savvides, H. Park, and M. Srivastava, "The Bits and Flops of the n-hop Multilateration Primitive for Node Localization Problems," *1st ACM Int'l Wksp. Wireless Sensor Networks and Apps.*, Atlanta, GA, Sept. 2002, pp. 112–21.
- [15] K. Whitehouse and D. Culler, "Macro-Calibration in Sensor/Actuator Networks," *Mobile Networks and Apps. J.*, Special Issue on Wireless Sensor Networks, vol. 8, no. 4, 2003, pp. 463–72.

BIOGRAPHIES

YING ZHANG (yzhang@parc.com) is a member of research staff at PARC. She received her Ph.D. from the University of British Columbia in computer science in 1994. Her current research interests are in sensor/actuator networks and embedded control architectures. She has led software

architectures for modular reconfigurable robots at PARC and has been working on constraint-based routing and localization on sensor networks.

MARK YIM (yim@grasp.upenn.edu) joined the University of Pennsylvania as an associate professor and Gabel Family Term Junior Professor in September 2004. Earlier he was a principal scientist at Xerox PARC. He received his mechanical engineering Ph.D. from Stanford University (1994) where he started his work on modular self-reconfigurable robotics, a rapidly growing research area. Recent honors include induction as a World Technology Network Fellow; election to Distinguished Lecturer for the IEEE Robotics and Automation Society; and induction into 1999 MIT's Technology Review TR100.

LEE ACKERSON (ackerson@parc.com) is an embedded systems engineer with 20 years of experience designing hardware and software for microcontrollers used in motor control, sensor networks, and communications. He holds patents in a wide variety of devices from regenerative braking for motor controllers to electric paper display technologies.

DAVID DUFF (dduff@parc.com) leads the efforts of the Smart Electro-Mechanical Systems group at PARC. During his six years at PARC he has designed several reconfigurable robots. He holds an M.S. in engineering from Stanford and a BSME from Ohio State University. Prior to joining PARC he worked for WET Design designing fountain equipment for the Bellagio fountain and as a systems engineer at Lockheed-Martin on strategic defense projects.

CRAIG ELDERSHAW (celdersh@parc.com) has studied at the University of Queensland, the Australian National University, and Stanford University. He received his Ph.D. in heuristic motion planning from Oxford University, United Kingdom, in 2001. Since then, he has worked with the Smart Electro-Mechanical Systems group at PARC. He has been involved in developing both the software and electronics in PARC's modular robotic systems.