SIMULTANEOUS LOCALIZATION AND TRACKING IN AN AD HOC SENSOR NETWORK

Christopher Taylor, Ali Rahimi, Jonathan Bachrach, and Howard Shrobe

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{taylorc, ali}@mit.edu, {jrb, hes}@csail.mit.edu

ABSTRACT

Make a note of similarity between SLAM and localization, tracking.

We demonstrate that it is possible to achieve accurate localization while target tracking in a randomly placed wireless sensor network composed of inexpensive components of limited accuracy. We present an algorithm for creating such a coordinate system without the use of global control, globally accessible beacon signals, or accurate estimates of inter-sensor distances. The coordinate system is robust and automatically adapts to the failure or addition of sensors. The algorithm learns from observations of local events, events that can be sensed in a particular neighborhood. Tracking improves over time providing a measurable error that can also be used to guide further exploration. The algorithm is based upon a general parameter estimation framework that easily incorporates a priori knowledge, provides error estimates on all measurements, and allows for well founded outlier rejection. Furthermore, this approach can be generalized to also simultaneously calibrate sensors and perform time synchronization and its coordinate system can optionally be aligned to surveyed positions.

1. INTRODUCTION AND FEATURES

Advances in technology have made it possible to build ad hoc sensor networks using inexpensive nodes consisting of a low power processor, a modest amount of memory, a wireless network transceiver and one or more sensors; a typical node is comparable in size to 2 AA batteries [1]. Many novel applications are emerging: habitat monitoring, smart building reporting failures, target tracking, etc. In these applications it is necessary to accurately orient the nodes with respect to the global coordinate system. Ad hoc sensor networks present novel tradeoffs in system design. On the one hand, the low cost of the nodes facilitates massive scale and highly parallel computation. On the other hand, each node is likely to have limited power, limited reliability, and only local communication with a modest number of neighbors. The application context and massive scale make it unrealistic to rely on careful placement or uniform arrangement of sensors. Rather than use globally accessible beacons or expensive GPS to localize each sensor, we would like the sensors to be able to self-organize a coordinate system.

In this paper, we present SLAT (Simultaneous Localization and Tracking), an algorithm that incrementally localizes the nodes in a sensor network as it tracks moving targets in real-time. Our method is based on the observation that each range measurement between a sensor and a target contains information about both the target's location and the sensor's position and that given a sufficient quantity of range data, we can simultaneously track a target and localize our sensors.

Using target ranges to perform localization provides several important advantages. The redundance inherent in the tracking measurement data provides resistance to individual measurement errors. Also, tracking accuracy improves in high-traffic areas, since these areas witness the most data. SLAT also allows networks to be localized without use of inter-node ranging, which allows sensors to be deployed without line of sight to each other (which is often desirable for ultrasound ranging systems [2]). SLAT provides all of these benefits in a highly extensible probabilistic framework.

SLAT localizes sensors into a coordinate system that spans the entire network. That means all sensors' position estimates are mutually comparable. If anchor nodes are present, SLAT can localize into an absolute coordinate system. In their absence, SLAT places sensors in a relative coordinate system that is correct up to a translation, rotation, and possible reflection. Though this paper focuses on localization when sensors and targets are located in a single plane, SLAT extends intuitively to three dimensional problems.

SLAT arises from a very extensible probabilistic framework. In the future, we will likely be able to cope with moving sensors. We also believe we can eventually calibrate the sensors' ranging hardware and perform time synchronization at the same time as we track and localize. Though will

not demonstrate these capabilities in this paper, we think our system is in a good position to include them in the future.

The next section introduces our approach from a high level. Section 3 explores our design in detail. Section 4 discusses our initial results. Sections 6 and 7 propose future work and describe related algorithms in the literature. We present our conclusions in section 8

2. SLAT OVERVIEW

In SLAT, sensors perceive a moving target as a sequence of discrete events. An event occurs when the target emits a tagged radio message followed by an acoustic pulse. Sensors can estimate distance by comparing the arrival times of the radio and acoustic signals using time difference of arrival (TDoA). An event occurs at the location of the target, and is detected nearly simultaneously by most sensors within range. These sensors are each able to compute an estimate of the distance between themselves and the event.

SLAT attempts to determine the most probable positions of the sensors and events given a collection of computed ranges by finding the posterior distribution over the sensors and events. To compute this distribution, SLAT begins by finding two other distributions: a prior distribution over the sensors and events, and a sensor model.

The prior distribution can be used to incorporate a variety of initial knowledge about the layout of the network. More importantly, it can retain knowledge about the sensor positions between groups of events. The prior in a sense is responsible for SLAT's learning behavior.

The sensor model is the probability distribution of range measurements given a particular placement of sensors and events.

SLAT is an iterative algorithm, since it processes events in small batches shortly after they occur. Each iteration begins with a simple Gaussian prior distribution over the sensors. When SLAT witnesses events, it computes the posterior over the positions of the sensors and events. It then generates a position estimate for the sensors and events based on the mode of the posterior. Finally, it reduces the posterior to a "prediction" that it approximates using a Gaussian distribution and reuses as the prior for the next group of events.

This process repeats continually. At each step, the sensor estimates become more accurate. Once these position estimates converge to the sensors' real positions, the SLAT algorithm becomes simple tracking. Figure 1 summarizes our architecture in diagram form.

Most of the work in each SLAT iteration goes into finding the mode of the posterior distribution. We perform this optimization using the Newton-Raphson method. However, like most optimization techniques, Newton-Raphson can converge to local optimums unless they are applied carefully.

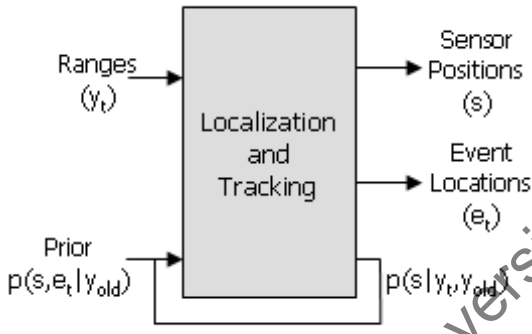


Fig. 1. The SLAT architecture. Given a Gaussian prior distribution and a collection of ranges to some events, SLAT produces an estimate of the sensors' positions, an estimate of the events' locations, and a more informative Gaussian prior for use with future computations.

SLAT chooses an initial prior distribution that helps the optimizer avoid these problems.

TODO: Ali needs to talk about EKF.

3. SLAT DETAILS

3.1. Definitions

For simplicity, we confine ourselves to two dimensions, though we can trivially extend our method to three. Let s_i be a two dimensional vector representing the position of the ith sensor, where $i = 1 \ldots n$, and define $s = \{s_i\}_{i=1\ldots n}$ to be the set of all sensor positions.

To name the events, we first split time into T discrete intervals. At the end of each interval, SLAT estimates positions of the events that occurred within the interval. Each interval contains several events. This is especially important when the prior distribution is based on few measurements. As more measurements are gathered, the interval size decreases to improve tracking speed. Let $e^t = \{e^t_j\}_{j=1\ldots m_t}$ be the set of events that occur during interval t, $t = 1 \ldots T$. e^t_j is the position of the jth event to occur in tth time interval.

Finally, let $y^t = \{y^t_{ij}\}$ be the set of sensor measurements taken during interval t. y^t_{ij} is the measurement taken by sensor s_i during the event e^t_j.

3.2. The posterior distribution

SLAT's design is focused on identifying and working with the posterior distribution, specifically:

$$p(s, e^t | y^1, y^2, \ldots, y^t) \quad (1)$$

That is, SLAT wants the distribution over the positions of the sensors and most recent events, given all measurements observed up until time interval t. At the end of each interval t, SLAT finds the distribution (1). It then produces a most probable estimate for s and e^t by finding the mode:

$$s^*, e^{t*} = \underset{s,e^t}{\text{argmax}}\ p(s, e^t | y^1, y^2, \ldots, y^t)$$

We will split the measurements $y^1 \ldots y^t$ into two groups. y^t is the most recent measurements, and $y_{old} = \{y_i\}_{i=1\ldots t-1}$ is the set of measurements incorporated by earlier iterations of the algorithm. The posterior $p(s, e^t | y^t, y_{old})$ can be expressed in terms of two important distributions: the prior distribution $p(s, e^t | y_{old})$, and the sensor model $p(y^t | s, e^t)$.

For notational convenience, define $x^t = \{s, e^t\}$, which makes the posterior $p(x^t | y^t, y_{old})$. By Bayes' Rule:

$$p(x^t | y^t, y_{old}) = \frac{p(y^t | x^t, y_{old}) p(x^t | y_{old})}{p(y^t | y_{old})} \tag{2}$$

Knowledge of measurements to earlier events does not provide any basis for predicting future measurements, so the new measurements in interval t (y^t) are independent of the measurement history prior to interval t (y_{old}). Equation (2) becomes:

$$\begin{aligned} p(x^t | y^t, y_{old}) &= \frac{p(y^t | x^t) p(x^t | y_{old})}{p(y^t)} \\ &\underset{x^t}{\propto} p(y^t | x^t) p(x^t | y_{old}) \end{aligned} \tag{3}$$

So as a function of the variables to estimate, the posterior distribution is proportional to the product of a "prior distribution" $p(x^t | y_{old}) = p(s, e^t | y_{old})$ and a "sensor model" $p(y^t | x^t) = p(y^t | s, e^t)$. The next two sections will discuss these two distributions in depth.

To summarize, here are the steps of the SLAT algorithm:

Step 1 Start with the prior distribution $p(x^t | y_{old})$.

Step 2 Observe new measurements y^t.

Step 3 Compute the posterior $p(x^t | y^t, y_{old})$ using the prior from step 1 and a sensor model $p(y^t | x^t)$.

Step 4 Compute the mode of the posterior, and report it as an estimate of the sensor positions s and event positions e^t.

Step 5 Compute the prediction $p(x^{t+1} | y^t, y_{old})$ using an approximation to the posterior.

Step 6 Return to step 1, using the prediction as the prior for interval $t + 1$.

3.3. The sensor model

First consider the sensor model $p(y^t | x^t)$. As defined in section 3.1, y^t is a set of measurements y^t_{ij}. Each measurement y^t_{ij} depends only on the sensor taking the measurement, s_i, and the event generating the measurement, e^t_j.

$$p(y^t | x^t) = \prod_{i,j} p(y^t_{ij} | s_i, e^t_j) \tag{4}$$

As we noted in section 2, we assume that the target produces a tagged radio pulse followed by a sonic pulse. This allows the sensors to easily differentiate between events, and allows the sensors to use time difference of arrival (TDoA) to obtain fairly accurate range measurements. We model these measurements as follows:

$$y^t_{ij} = \| s_i - e^t_j \| + \omega^t_{ij} \tag{5}$$

$\| \cdot \|$ indicates the vector 2-norm, which is the same as the Euclidian distance between s_i and e^t_j. ω^t_{ij} represents some amount of zero-mean Gaussian noise with variance σ^2. We chose this model because it is similar to noise models used in the literature [3, 4], and because it is mathematically tractable. We could also have used more sophisticated modelling techniques – in particular, we could use a particle representation. TODO: Ali, talk about particle representations.

From (5), we can directly state $p(y^t_{ij} | s_i, e^t_j)$ – it is a univariate Gaussian with mean $\| s_i - e^t_j \|$ and some variance σ^2:

$$p(y^t_{ij} | s_i, e^t_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\| s_i - e^t_j \| - y^t_{ij})^2}{2\sigma^2} \right] \tag{6}$$

In the absence of an informative prior distribution $p(x^t | y_{old})$, equations (3), (4), and (6) reduce to the simple squared measurement error metric often used in the literature, for instance in "spring model" approaches such as [4]. However, the use of an informative prior makes our algorithm substantially more capable. TODO: fix this

3.4. The prior distribution

The prior distribution $p(x^t | y_{old})$ encodes an a priori belief about the relative likelihood of different sensor and event locations. It encapsulates two different distributions: $p(s | y_{old})$ and $p(e^t)$. The product of these distributions is the prior $p(x^t | y_{old})$, since e^t is independent of both y_{old} and s.

We require that the prior distribution $p(x^t | y_{old})$ be a multivariate Gaussian so that the product of the prior and the sensor model defined in the last section is manageable. This in turn suggests that $p(s | y_{old})$ and $p(e^t)$ also be Gaussian.

$p(s | y_{old})$ has two forms, which we will summarize here:

- *System startup.* In this case, there are no measurements in y_{old}. We create a Gaussian prior $p(s)$ encodes any information we have about the sensor locations. This information might come from specialized hardware (e.g. GPS) or other localization techniques. In particular, we use the results of a lower precision localization algorithm based on radio signal strength or radio hop count as a weak prior to help SLAT avoid false optima. We discuss specific algorithms and the local minimum problem in section 3.7.

- *Online updates.* In the update case, SLAT has a nontrivial measurement history y_{old}. This implies that SLAT has already computed the distribution $p(x^{t-1}|y^{t-1} \ldots y^1) = p(s, e^{t-1}|y_{old})$. We simply marginalize out the events e^{t-1} to get $p(s|y_{old})$. We then approximate $p(s|y_{old})$ with a multivariate Gaussian $q(s|y_{old})$ using Laplace's method. $q(s|y_{old})$ becomes the basis for a "prediction" for interval t: $p(x^t|y_{old})$. We show how to derive this prediction from the posterior in section 3.6.

SLAT also uses a prior on the events, $p(e^t)$, which we show how to compute in section 3.7. Like $p(s)$, $p(e^t)$ primarily helps SLAT avoid trouble with local optimums.

The SLAT prior consists of two multivariate Gaussians. $p(s|y_{old})$ is described by its mean s_0^t and its covariance Λ_s^t. Likewise, $p(e^t)$ is defined by its mean e_0^t and its covariance Λ_e^t. As a result, the algebraic form of $p(x^t|y_{old})$ is as follows, where Z is a normalization constant:

$$\begin{aligned} p(x^t|y_{old}) &= \frac{1}{Z}\exp\left[-\frac{1}{2}(x^t - x_0^t)^T[\Lambda_x^t]^{-1}(x^t - x_0^t)\right] \\ x_0^t &= \begin{bmatrix} s_0^t \\ e_0^t \end{bmatrix} \\ \Lambda_x^t &= \begin{bmatrix} \Lambda_s^t & 0 \\ 0 & \Lambda_e^t \end{bmatrix} \end{aligned} \tag{7}$$

$p(x^t|y_{old})$ is the prior or prediction we require to compute the posterior distribution $p(x^t|y^t, y_{old})$. We will return to specific choices of Gaussians for $p(s|y_{old})$ and $p(e^t)$ in sections 3.6 and 3.7.

3.5. Finding the mode of the posterior

In this section, we show how SLAT computes a specific set of sensor and event positions from the posterior distribution. As we mentioned in section 2, SLAT computes this estimate x^{t*} by finding the mode of the posterior:

$$\begin{aligned} x^{t*} &= \underset{x^t}{\text{argmax}}\; p(x^t|y^t, y_{old}) \\ &= \underset{x^t}{\text{argmin}}\; -\log p(x^t|y^t, y_{old}) \\ &= \underset{x^t}{\text{argmin}}\; \Big[(x^t - x_0^t)^T[\Lambda_x^t]^{-1}(x^t - x_0^t) \\ &\qquad + \frac{1}{\sigma^2}\sum_{i,j}(\|s_i - e_j^t\| - y_{ij}^t)^2\Big] \end{aligned} \tag{8}$$

We can now optimize (8) using the Newton-Raphson method. The Newton-Raphson method is an iterative technique for solving minimization problems. Starting at some point x_i^t, Newton-Raphson fits the error surface (8) with a quadratic whose value, gradient, and Hessian match those of the error surface at x_i^t. It then places x_{i+1}^t at the bottom of the quadratic. The process continues until it converges at the minimum of the error surface. In principle, Newton-Raphson is similar to gradient descent, but boasts a faster rate of convergence.

We show in section 9 that the update equation for Newton-Raphson on our problem is:

$$x_{i+1}^t = (A^T A + [\Lambda_x^t]^{-1})^{-1}(A^T b + [\Lambda_x^t]^{-1}x_0^t) \tag{9}$$

A and b are defined in section 9 as a matrix and vector respectively. Both are computed using x_i^t. This update equation gives us a complete method for finding the mode of the posterior $p(x^t|y^t, y_{old})$.

1. Set the starting point x_0^t to be the mean of the prior distribution $p(x^t|y_{old})$.
2. Repeat until the x_i^t's converge:

2a. Compute the matrix $Z = [\Lambda_x^t]^{-1} + A^T A$ and the vector $y = [\Lambda_x^t]^{-1}x_0^t + A^T b$.

2b. Solve the linear system $Zx_{i+1}^t = y$.

3. The final x_i^t is x^{t*}, the mode of $p(x^t|y^t, y_{old})$.

This is the most important part of the SLAT algorithm. Given a multivariate Gaussian prior $p(x^t|y_{old})$ and a quantity of range data y^t, SLAT produces the most probable sensor topology and event locations.

In the next two sections, we turn to the question of how to create the prior $p(x^t|y_{old}) = p(s|y_{old})p(e^t)$ in order to maximize SLAT's effectiveness.

3.6. Computing a prediction from the posterior

The most important use of the prior $p(x^t|y_{old})$ is encapsulating knowledge of the sensor positions s based on old measurements. To do this, we first find $p(s|y_{old})$.

$$p(s|y_{old}) = \int p(s, e^{t-1}|y^{t-1}, \ldots, y^2, y^1) de^{t-1} \quad (10)$$

Unfortunately, (10) is hardly a simple integral. It also seems unlikely that $p(s|y_{old})$ is Gaussian. We need $p(s|y_{old})$ to be Gaussian, since Gaussians are compact (since they are represented by a single vector of means and a single matrix of covariance) and easy to work with.

We resolve both problems by simply approximating the posterior with a Gaussian using Laplace's method. We will call this Gaussian distribution $q(s, e^{t-1}|y_{old})$. Since we are using Laplace's method, the mean q^* is the mode of the posterior $x^{(t-1)*}$, which we computed in section 3.5. The inverse covariance Λ_q^{-1} is the Hessian of the negative log posterior:

$$\begin{aligned} \Lambda_q^{-1} &= -\nabla^2 \log p(x^{t-1}|y_{old})|_{x^{t-1}=x^{(t-1)*}} \\ &= A^T A + [\Lambda_x^{t-1}]^{-1} \end{aligned} \quad (11)$$

The derivation of this formula is presented in appendix 9. Next, we integrate $q(s, e^{t-1}|y_{old})$ with respect to e^{t-1} to get the multivariate Gaussian prior $q(s|y_{old})$. This integration can be performed simply by dropping the rows and columns of Λ_q and the elements of the mean q^* that correspond to the events e^{t-1}, to produce the required covariance Λ_s^t and mean s_0^t.

$q(s|y_{old})$ is exactly the Gaussian prior we want. We can now form $p(x^t|y_{old})$, and use it to compute a posterior for interval t. Note that as we incorporate new measurements after each time interval, the storage used to maintain SLAT state remains constant – just s_0^t and Λ_s^t. This allows SLAT to handle large amounts of data over the life of the network.

The ability to save state using a prior also means that the localization of the network can converge over time. Eventually, the prior $p(x^t|y_{old})$ dominates the term $p(y^t|x^t)$ in the optimizer. This leads to little change in the position estimate s^*. It also leads to quick convergence in the optimizer, since the optimization begins very close to x^*. We expect SLAT's performance to increase over time, both in accuracy and in speed. Our simulation results (which we present in section 4) support this claim.

3.7. Avoiding local minima using a weak prior

Localization systems that attempt to localize using an optimization over range measurements often encounter trouble with local minima. As shown by Priyantha et al. [4], most of these problems stem from topological "folds." Figure 2 illustrates the problem graphically. Essentially, the possibility of folds causes concavities in the error surface analyzed by the optimizer.

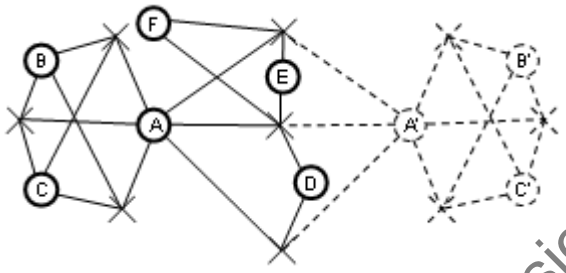


Fig. 2. An example of topology folding. Note that range estimates (shown as edges in the graph) cannot differentiate between the placement (A, B, C) and the placement (A', B', C'). As a result, the error surface for optimization is bi-modal. If D, E, and F have known positions, then without additional information to differentiate the between two modes, the optimizer has an unacceptably high chance of choosing the wrong mode. The wrong mode results in A, B, and C suffering grevious error relative to D, E, and F. Typically, one or two such folding errors leads to wildly inaccurate topology estimates.

In SLAT, we use an initial prior $p(s)$ to avoid the vast majority of these problems. There are several localization algorithms in the literature that are (1) simple, (2) distributed, and (3) hop-count based. These algorithms are typically somewhat inaccurate, since their only source of data is the radio connectivity of the network. However, several of these, such as the unfolding phase of [4] and the gradient multilateration approach of [5] and [6] completely avoid the issue of local minima, since they are not based on an optimization.

SLAT can use any of these algorithms to generate a fold-free topology of the sensor network, as in [4]. This topology's accuracy may be bad, but it does provide a total order over the sensors – that is, each sensor in roughly the right part of the plane relative to the rest of the sensors. By encoding this total order as a very weak prior, SLAT biases the optimizer in favor of the correct minimum. For instance, the fold-free topology might place A in figure 2 near F. This allows the optimizer to correctly choose the positioning (A, B, C) over the incorrect positioning (A', B', C').

The mathematics of this weak prior $p(s)$ are simple. Suppose an algorithm such as the unfolding phase from [4] generates positions $s_{unfolded}$. Then, the mean s_0 of $p(s)$ is simply $s_{unfolded}$. The covariance matrix Λ_s is wI, where w is a large number we choose (typically $w \approx 10^8$), and I is the identity matrix.

It is also helpful to provide a prior $p(e^t)$ that gives the events a good starting point. This helps avoid local minima and speeds convergence. SLAT generates a good placement quickly using the sensor means s_0 produced after the last

time interval. For each event, it chooses the three sensors whose ranges to the event are smallest. It then simply averages the position estimates of the sensors to get an estimate of the event's position. The covariance of $p(e^t)$ is also set to wI, where w is again a large constant.

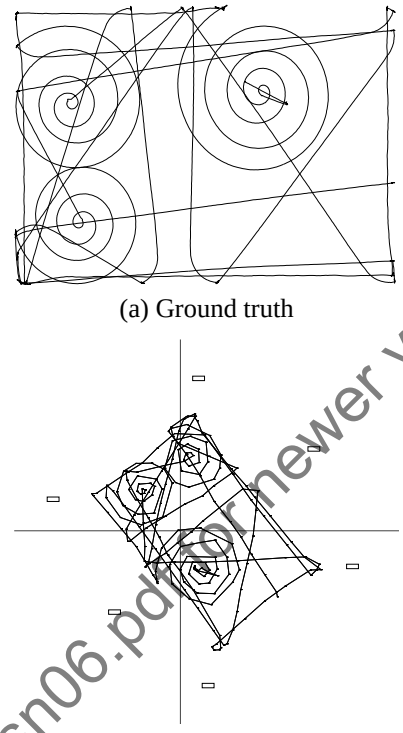
This initial prior $p(e^t)$ can be combined with $p(s|y_{old})$ or $p(s)$ to form $p(x|y_{old})$, the full prior.

4. RESULTS

4.1. Cricket

In our first experiment, a roomba is placed in a rectangular wooden enclosure. A cricket [CITE] is attached to the roomba [CITE]. This cricket beacons periodically using its radio and ultrasound transmitter. Six more crickets were placed outside the enclosure facing the roomba and instructed to listen for beacon pulses. The roomba was turned on and allowed to move freely within the enclosure. The roomba's movements were tracked using a video camera.

The ranges computed by the six external crickets were collected and processed in a Java-based simulator. This simulator simply applied the SLAT algorithm to the measurement data. The resulting target path and sensor positions are plotted in figure 3. Figure 3 also contains the target path extrapolated from the video camera gootage for comparison. The SLAT algorithm produced a rather high fidelity reconstruction of the roomba's path. It also correctly identified the placement of the six cricket sensors.



(a) Ground truth

(b) SLAT estimate

Fig. 3. Ground truth and SLAT estimate for Roomba experiment. Note that the SLAT estimate is affected by a rigid transformation.

4.2. Decreasing error over time

The remainder of our results are based on simulated sensor network topologies. Sensor placements are made at random, with the constraint that sensors be scattered fairly evenly in a rectangular area. Events are placed on random continuous paths through the sensor network's area. Sensor-event ranges are simulated based on the characteristics of Maroti et al's acoustic ranging system [CITE] for the Berkeley mica2 mote [CITE]. These measurements have a standard deviation error of about 8 cm.

Figure 4 shows a representative plot of SLAT estimate error versus the number of events witnessed by the network. In this network, 60 sensors were scattered over a 600 cm x 600 cm area. The average error in estimated sensor positions falls off rapidly at the beginning, and then slowly declines below the average measurement error. The initial falloff occurs since each measurement taken provides significant new information. This causes the estimate to change quickly. Once enough measurements have been taken, further refinement only occurs due to additional measurements cancelling out each others' noise. This accounts for the gradual improvement encountered after the initial drop.

The average event estimate error declines somewhat in the first few SLAT iterations, and soon stabilizes around the average measurement error. This is entirely reasonable, since it indicates that the majority of the error in an event estimate is the noise in the measurements to that event.

Figure 5 plots sensor estimate error against the number of measurements taken by the sensor. Note that all three quantities (minimum, average, and maximum error) decline as more measurements are taken. This indicates that as expected, SLAT's accuracy increases over time, and increases fastest in the areas with highest traffic.

Figure 6 shows the simulated sensor network used in figures 4, 5, and 7 at several points in time. These pictures show the SLAT process converging over time to a highly accurate localization estimate while reliably tracking targets.

Figure 7 shows the average variance of the estimated positions in the SLAT posterior after each time interval. The variance of sensor position estimates starts very large, and rapidly drops as measurements are incorporated. This corresponds to increased confidence in the SLAT estimates. As expected, the event variances are relatively constant.

We claimed that the Newton-Raphson method of maximizing the posterior distribution converges quickly. Em-

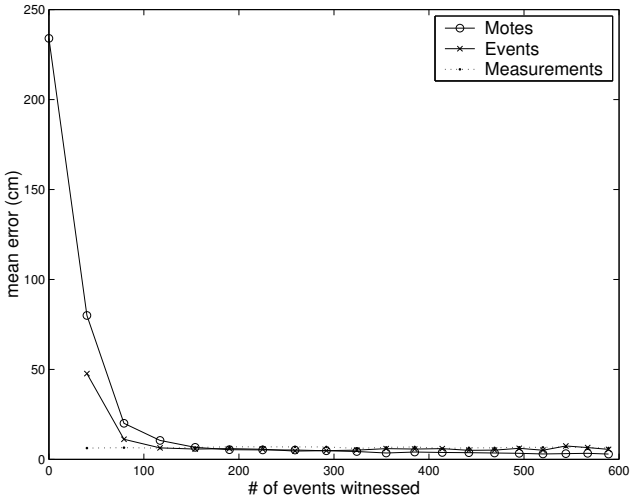


Fig. 4. SLAT estimate error vs. number of observed events. In this simulation, 60 sensors are spread over a 600 cm x 600 cm area. Simulated targets move along continuous paths through the network producing events as they move. Each mark on the graph represents the status of the network after an iteration of the SLAT algorithm. Error is measured relative to ground truth. As predicted, SLAT's localization estimate improves over time; this drives the improvement in event error.

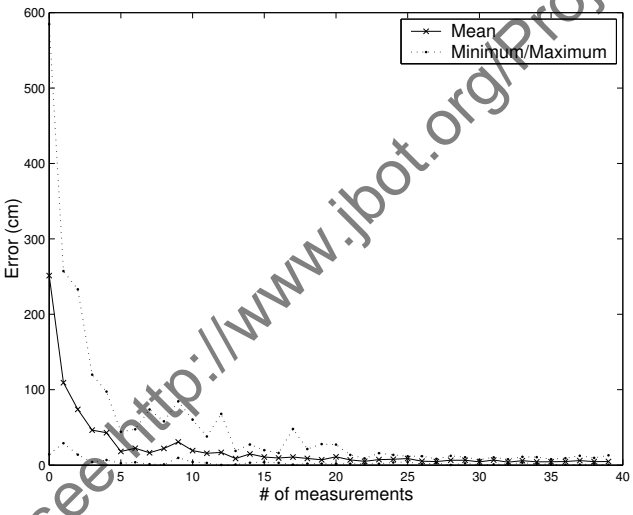
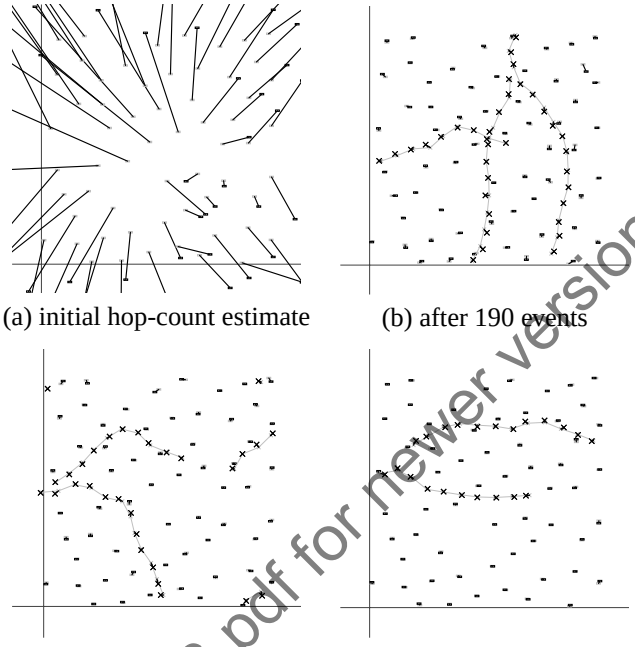


Fig. 5. SLAT sensor estimate error vs. number of observed measurements per sensor for the simulation in figure 4. The three plotted lines indicate the maximum, average, and minimum error in the SLAT estimate.



(a) initial hop-count estimate (b) after 190 events

(c) after 385 events (d) after 589 events

Fig. 6. These four graphs show a simulated sensor network at several points in time. The estimates are dark colored. Lines indicate the correspondence between estimates and ground truth positions. Figures b-d show the event estimates for the most recent time interval as 'x' marks.

pirically, each SLAT iteration typically requires on average 3.25 Newton-Raphson iterations to converge.

4.3. Sensitivity to measurement error

Figure 8 demonstrates the effect of increased measurement error on SLAT's accuracy. We ran simulated experiments with a measurement error standard deviations of 8 cm, 14 cm, and 20 cm. Each increase in variance slightly increases the number of events required before the initial decrease in sensor position estimate error is complete. Furthermore, each increase in measurement error slightly increases the asymptotic amount of measurement error.

4.4. Sensitivity to density

Figure 9 shows the effect of sensor density on SLAT effectiveness. Each decrease in density means that fewer sensors witness any particular event. Intuitively, this suggests that less dense networks will improve less quickly. Furthermore, we know there is some threshold of density beyond which SLAT will not converge at all, since at least three measurements are required simply to track a target. Both of these intuitions are supported by simulation.

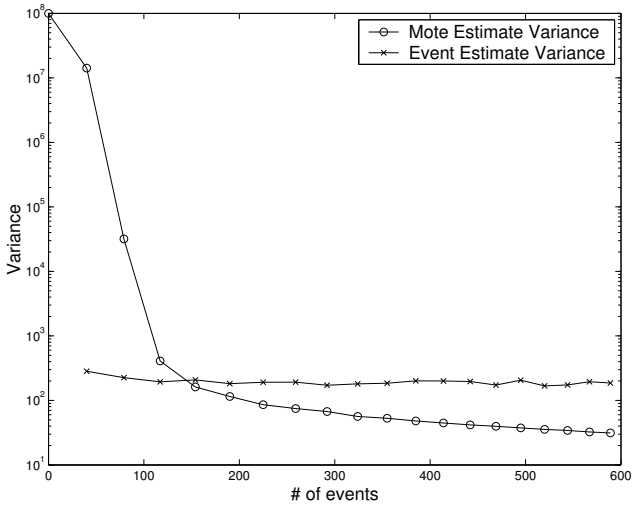


Fig. 7. SLAT prediction variance vs. number of observed events for the simulation in figure 4. The plotted quantity is the average of the values on the diagonal of Λ_s^t and Λ_e^t.

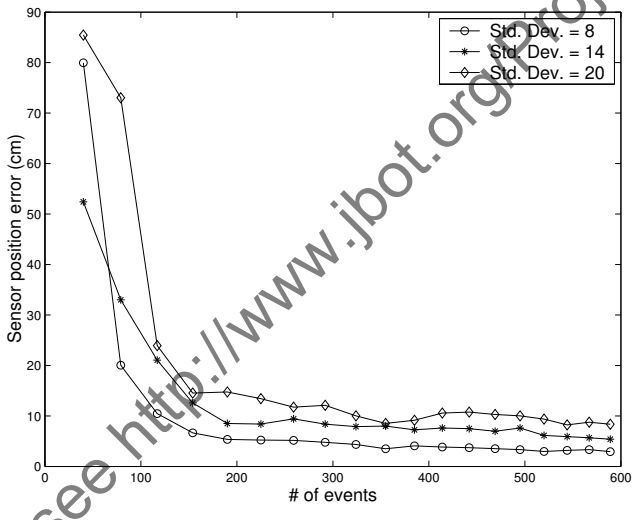


Fig. 8. Effect of measurement error on SLAT localization accuracy.

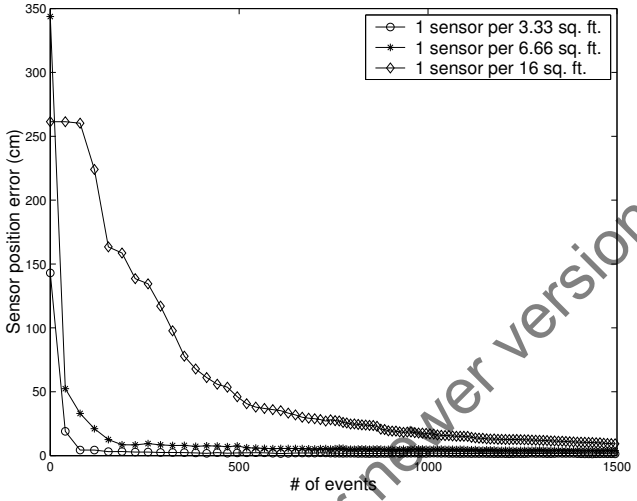


Fig. 9. Effect of sensor density on SLAT localization accuracy.

4.5. Performance on larger networks

Figure 10 demonstrates the SLAT algorithm on a larger network. This larger network maintains a constant density, but contains four times as many sensors. The decrease in convergence speed is easily explained. As we showed in figure 5, sensor estimate accuracy is primarily dictated by the number of measurements observed. This suggests that a larger network with identical density will require more events in order to provide the requisite number of measurements for each sensor. This turns out to be the case.

Some of the effect is also caused by our simulator's algorithm for generating target paths. These paths tend to avoid the corners of the network, which results in some sensors improving more slowly than others. As we noted earlier, this behavior is expected and acceptable, since the slowly improving sensors are always in low traffic areas.

5. DISTRIBUTED SLAT

In this paper, we have primarily presented SLAT as a centralized algorithm. However, we have every reason to believe our techniques can be implemented to run in distributedly inside the network using only local communication between nodes. In fact, we expect the space and time requirements of distributed SLAT to be $O(n_{local})$, where n_{local} is the number of nodes within two times the sensing range of a node. This is possible because the important matrices in SLAT, $A^T A$ and Λ_t^{-1}, contain internal structure which we can exploit for efficient storage in the network.

Furthermore, the heavy lifting computation in SLAT, the solving of a large system of linear equation in (9), can be

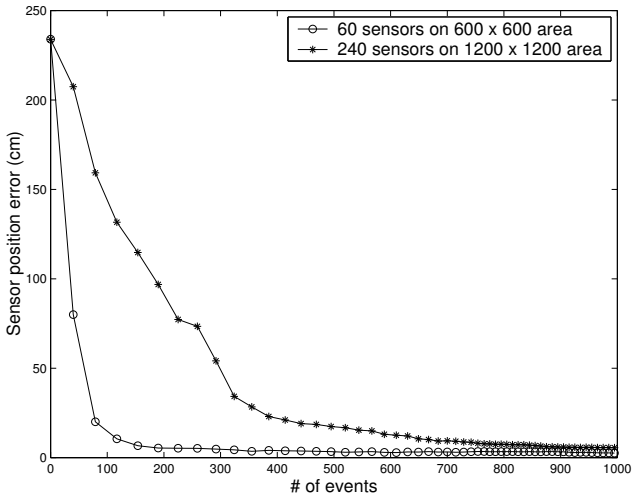


Fig. 10. Effect of network size on SLAT localization accuracy.

distributed using a type of Richardson Iterations [7] known as the Gauss-Siedel method. Distributed least squares solvers have been used in sensor networks in the past. Jacobi iterations in particular are quite common [4]. These techniques never require the entire linear system to be assembled on a single processor; in fact, their computation patterns integrate nicely with the storage scheme dictated by the structure of $A^T A$ and Λ_t^{-1}. The task of applying Laplace's method to the posterior and marginalizing events (section 3.6) has a closed form that is amenable to distributed computation.

We are optimistic that we can implement SLAT in a distributed fashion.

6. FUTURE WORK

As we saw in section 4, sensor calibration can make a substantial difference to both localization and tracking accuracy. We believe that we can detect the mean measurement error within the SLAT parameter estimation framework. This would allow sensor calibration to occur concurrently with localization and tracking.

Events are rarely randomly located. Targets follow a somewhat predictable path based on the physics of their movement. We think that by applying a special dynamics prior to the events in our estimation framework, we can further improve our target tracking accuracy. It is also likely that target dynamics will assist us in avoiding local minima for event estimates.

Our framework may also be capable of accomodating mobile sensors. By applying dynamics to the prior $p(s|y_{old})$, we may be able to detect changes in sensor location.

SLAT currently depends on TDoA range measurements from sensors to targets. We think it may be possible to localize sensors based solely on common observations of environmental noise. We would estimate relative distance to the target by comparing the arrival times of a single sound at different sensors. The goal would then be to locate the source of the sound while localizing the sensors. It may even be possible to leverage the information from the sound arrival times to perform time synchronization between sensors.

Finally, we hope to complete a true distributed implementation of the SLAT algorithm on real sensor network hardware.

7. RELATED WORK

I think I will need Ali's help writing about SLAM research. We should talk about Sebastian Thrun's work, and Phil McLauchlan's work on VSDF.

In the sensor network space, there are no papers to our knowledge that address simultaneous localization and target tracking. The closest algorithms to SLAT are those that perform some type of optimization, such as multidimensional scaling by Shang et al. and Ji et al. [8, 9], linear/semidefinite programming by [10], and the algorithms that apply squared error minimization techniques [4, 3]. SLAT's probabilistic modeling makes it substantially more powerful and sophisticated than these algorithms.

Our technique for local minimum avoidance in section 3.7 is inspired primarily by the work of [4], which explores the importance of fold freedom when localizing a network using squared error metrics. In fact, as we stated in section 3.7, we use the unfolding technique presented in their paper as SLAT's initial prior. We also suspect that the gradient multilateration approach presented by Nagpal et al. [5] produces a good SLAT prior, though we have not performed any experimental validation.

Moore et al. [3] have also done interesting work in local minimum avoidance, specifically on the topic of detecting flip-prone topologies. This enables their algorithm to avoid making poor choices when facing multi-modal error surfaces. We are considering adapting this technique to help detect the occasional mistakes SLAT makes at the edges of sensor topologies.

8. CONCLUSION

Hit the high points one more time. Yell with triumph about our results. Emphasize that our approach is deeply unique: none of the related work we know does this stuff. It is awesome. I am punting on this until later. ———— In this paper, we presented an algorithm for simultaneously constructing a coordinate system and tracking targets from only

observation of common events. Our algorithm relies on simple distributed computation and local communication only, features that an ad hoc sensor network can provide in abundance. At the same time it is able to achieve very reasonable accuracy. The algorithm gracefully adapts to take advantage of any improved sensor capabilities or availability of additional seeds.

9. APPENDIX: NEWTON-RAPHSON

In this appendix, we show how to fit the negative log posterior (equation (8)) with a quadratic function. We then show that the minimum of this quadratic yields the Newton-Raphson update formula (9). Finally, we compute the Hessian of the negative log posterior for use in Laplace's method, in section 3.6 (equation (11)).

(8) has two parts: a quadratic due to the multivariate Gaussian prior, and a sum of squared non-linear functions from the sensor model. We fit the latter term with a quadratic, from which we can trivially construct a quadratic that fits (8).

It turns out that when the error surface can be expressed as a sum of squares, it is possible to determine the fitting quadratic without explicitly calculating the second derivative. In mathematical terms, suppose the error surface is:

$$\sum_{i,j}(f_{ij}(x^t))^2 \tag{12}$$

We can find a fitting quadratic by replacing $f_{ij}(x^t)$ with its first-order Taylor series around x_i^t. So the quadratic that fits (12) is:

$$\begin{aligned}&\sum_{i,j}(f_{ij}(x_i^t)+\nabla f_{ij}(x_i^t)(x^t-x_i^t))^2\\ =&\sum_{i,j}([\nabla f_{ij}(x_i^t)]x^t-[-f_{ij}(x_i^t)+\nabla f_{ij}(x_i^t)x_i^t])^2\end{aligned} \tag{13}$$

$$f_{ij}(x^t) = \|s_i - e_j^t\| - y_{ij}^t \tag{14}$$

Equation (13) can be rewritten as a least squares problem by stacking terms:

$$\begin{aligned}&\sum_{i,j}([\nabla f_{ij}(x_i^t)]x-[-f_{ij}(x_i^t)+\nabla f_{ij}(x_i^t)x_i^t])^2\\ =&\|Ax^t-b\|^2\end{aligned} \tag{15}$$

$$A=\begin{bmatrix}\nabla f_{00}(x_i)\\ \vdots\\ \nabla f_{nm}(x_i)\end{bmatrix}$$

$$b=\begin{bmatrix}-f_{00}(x_i)+\nabla f_{00}(x_i)x_i\\ \vdots\\ -f_{nm}(x_i)+\nabla f_{nm}(x_i)x_i\end{bmatrix}$$

Thus, the quadratic that fits (8) at x_i^t is:

$$0=(x^t-x_0^t)^T[\Lambda_x^t]^{-1}(x^t-x_0^t)+\|Ax^t-b\|^2 \tag{16}$$

To complete Newton-Raphson we set x_{i+1} to be the minimum of (16):

$$x_{i+1}=\operatorname*{argmin}_x (x-x_0)^T\Lambda_x(x-x_0)+\|Ax-b\|^2$$

We can do this in closed form by setting the gradient of the quadratic to zero and solving, since the gradient is zero only at the extremum of a quadratic.

$$\begin{aligned}0&=\nabla\left[(x^t-x_0^t)^T[\Lambda_x^t]^{-1}(x^t-x_0^t)+\|Ax^t-b\|^2\right]\\ &=2[\Lambda_x^t]^{-1}x^t-2[\Lambda_x^t]^{-1}x_0^t)+2A^TA-2A^Tb\end{aligned} \tag{17}$$

$$([\Lambda_x^t]^{-1}+A^TA)x^t=[\Lambda_x^t]^{-1}x_0^t+A^Tb \tag{18}$$

Equation (18) is the update formula we require.

We conclude the appendix by computing the Hessian of (8), which we used in section 3.6. This is simple, since (17) contains the gradient: we simply apply another gradient:

$$-\nabla^2\log p(x^t|y^t,y_{old})|_{x^t=x_i^t}=[\Lambda_x^t]^{-1}+A^TA \tag{19}$$

10. ACKNOWLEDGEMENTS

This research is supported by DARPA under contract number F33615-01-C-1896.

11. REFERENCES

[1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proceedings of ASPLOS-IX*, 2000.

[2] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan, "The cricket location-support system," in *Proceedings of MobiCom 2000*, August 2000.

[3] David Moore, John Leonard, Daniela Rus, and Seth Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of ACM Sensys-04*, Nov 2004.

[4] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," 2003.

[5] R. Nagpal, "Organizing a global coordinate system from local information on an amorphous computer," 1999.

[6] C. Savarese, J. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," 2001.

[7] G. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1989.

[8] Shang, Ruml, Zhang, and Fromherz, "Localization from mere connectivity," in *MobiHoc*, 2003.

[9] X. Ji and H. Zha, "Sensor positioning in wireless ad hoc networks using multidimensional scaling," in *Infocom*, 2004.

[10] L. Doherty, L. El Ghaoui, and K. S. J. Pister, "Convex position estimation in wireless sensor networks," in *Proceedings of Infocom 2001*, April 2001.