Macro-calibration in Sensor/Actuator Networks

Kamin Whitehouse
Computer Science
UC Berkeley
Berkeley, CA 94720

David Culler
Computer Science
UC Berkeley
Berkeley, CA 94720

Abstract

We describe an ad-hoc localization system for sensor networks and explain why traditional calibration methods are inadequate for this system. Building upon previous work, we frame calibration as a parameter estimation problem; we parameterize each device and choose the values of those parameters that optimize the overall system performance. This method reduces our average error from 74.6% without calibration to 10.1%. We propose ways to expand this technique to a method of autocalibration for localization as well as to other sensor network applications.

1 Introduction

Sensor networks present new challenges in calibration. Many sensors today have an easy, built-in calibration interface, such as police traffic radars that use a tuning fork [4]. Others are factory calibrated and built to remain calibrated for long periods of time, such as industrial quality accelerometers [5]. Sensor systems have also traditionally used only a small number of sensors. In robotics, for example, even "Multi-sensor" systems consist of only a handful of sensors [19]. Even currently deployed sensor "networks", such as the SeaKeepers Society Ocean Monitoring System, are of the *macro* variety, where each sensor is individually managed [13].

As a result of this, most sensor-systems are built for *micro*-calibration, in which each device is individually tuned in a carefully controlled environment.

Sensor networks, however, demand a new method of calibration. Individual calibration of hundreds or thousands of devices can be problematic, especially when many small or low-power devices have no calibration interface. Furthermore, devices often need to be calibrated in partially unobservable and dynamic environments, or may even be unobservable themselves. They are also often general-purpose devices that need to be calibrated differently for each application.

The demand for *macro*-calibration has become real in our ad-hoc localization system for sensor networks. The devices we use are so integrated into the system that we cannot easily observe them for calibration. Even if they could be observed, there are too many devices to manage each one individually. They do not have calibration interfaces and even if they did it would be unclear how to use them to calibrate, for example, an acoustic device for distance estimation.

We explore macro-calibration by framing calibration as a general *parameter estimation* problem. For each device, we choose calibration parameters that optimize the overall system response, instead of the individual device responses. There are many benefits to this technique. First, it frees us from the need to directly observe and calibrate each and every device; we only need to observe the overall system response. Second, it provides a calibration interface to those devices that do not already have one. This interface is specific to our application instead of specific to the device. Finally, it suggests a method of autocalibrating distance estimates without actually knowing the true distances between the nodes.

The rest of this paper is organized as follows. Section 2 introduces our ad-hoc localization system and its calibration problems. Section 3 formalizes calibration in the traditional sense and explains why it is inadequate for localization. Section 4 discusses two micro-calibration techniques for localization while section 5 describes a method of macro-calibration. Sections 6 and 7 evaluate and compare these three methods. Section 8 describes an extension of this calibration technique to an autocalibration method for localization. Section 9 suggests ways to extend this work to other sensor network application.

2 Ad-hoc Localization and the Calibration Problem

Localization becomes a much more difficult problem in the context of ad-hoc sensor networks. Almost all localization systems existing today rely heavily on *infrastructure* to provide known positions and distances. GPS and Active Bats from AT&T use the known positions to anchor the system in coordinate space [12] [8]. Cricket from MIT uses the known distances to automatically calibrate for temperature and humidity, which affect the speed of sound [16]. Some localization systems also leverage deployment time to help with calibration. For example, Microsoft's RADAR [1] can use location fingerprinting, which is the process of characterizing every location in the environment for effects such as RF attenuation. Many other localization systems also rely heavily on specialized hardware. GPS for example uses atomic clocks for time synchronization while Active Bats uses an array of ultrasonic receivers.

In contrast, ad-hoc localization systems have no infrastructure to provide known distances or positions. They also cannot rely on having any deployment time and are limited to using small, low-power components.

To investigate ad-hoc localization, we have built an ad-hoc localization system called *Calamari* [17] that aims to consume as few resources as possible, including energy, computational power, and componentry. Calamari estimates the distance between nodes using a fusion of RF received signal strength information (RSSI) and acoustic time of flight (TOF), both of which are relatively cheap in all regards.

In RSSI ranging, one node transmits a clean RF carrier frequency of 916.5MHz and another node samples the received signal strength. Since the nodes already have radios for communication, this is a low-power method of ranging at almost no computational cost that requires no additional hardware.

TOF ranging in Calamari uses more power and requires special hardware but yeilds more accurate distance estimates than RSSI. The transmitter sends short simultaneous RF and acoustic pulses while the receiver compares the time of arrival of both pulses. Since light and sound travel at different speeds, the time difference on arrival (TDOA) reveals the distance between the transmitter and receiver. By using a completely analog solution instead of digital signal processing, this normally

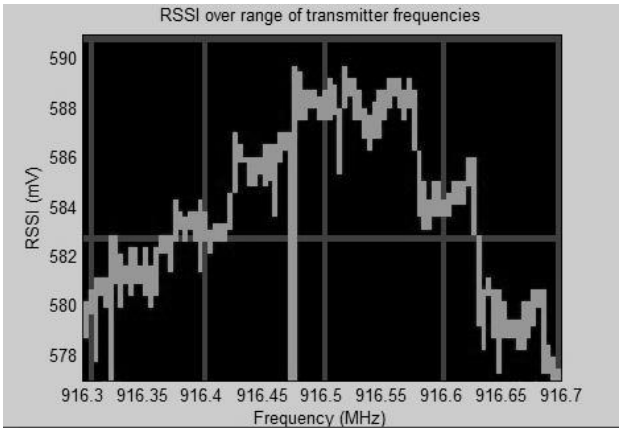

Figure 1: **Received Signal Strength over Frequency**. *All transmitters have slightly different frequencies, which will affect received signal strength readings as seen here.*

computationally intense ranging method is now virtually free.

The mass-produced, analog components used in Calamari provide a cheap, low-power solution but also introduce high variability between nodes, which means that we need sophisticated calibration in order to obtain reasonable results.

For example, a radio may transmit at up to twice the power of another radio, leading to distance errors of up to 100%. Variations in transmitter frequency also affect the observed RSSI, as seen in Figure 1 which shows the RSSI values of one receiver as the transmission frequency was varied over the range of transmitter frequencies. The variations in acoustic hardware are similar. With all the different types of hardware variation, the distance estimates from two different transmitter/receiver pairs can vary by as much as 300%, as seen in Figure 4.

These figures are representative of the tradeoff between needing to heavily engineer a system and needing to heavily calibrate it. The rest of this paper shows how we solve these calibration problems instead of adding extra specialized hardware, using expensive digital signal processing, or adding infrastructure. Unfortunately, as we see in the next section, traditional calibration is not an adequate solution and we need to find more powerful techniques.

3 Traditional Calibration

Device calibration is the process of forcing a device to conform to a given input/output mapping. This is often done by adjusting the device internally but can equivalently be done by passing the device's output through a *calibration function* that maps the actual device response to the desired response. In this paper, we will focus on the latter method because it is easily compared to the methods used in macro-calibration.

A more formal description of calibration is this: each device has a set of parameters $\beta \in \Re^p$. The purpose of calibration is to choose the correct parameters for each device such that they, in conjunction with a calibration function, will translate any actual device output r into the corresponding desired output r^*. The calibration function must therefore be of the form

$$r^* = f(r, \beta) \tag{1}$$

In *traditional calibration*, we observe the device in a controlled environment and map its observed output r to the desired output r^*, perhaps using a form of regression or even simply a lookup table.

To see why traditional calibration is inadequate for Calamari, let us try to use it to calibrate our ranging system. Assume we have a transmitter and receiver that give distance estimates d_1, d_2, and d_3 at known distances d_1^*, d_2^*, d_3^*. According to the traditional calibration framework, we need to generate a function with parameters β that map the estimates to the true distances. Let us try doing this with linear regression. First, we assume that r^* is a linear function of r, giving us the following calibration function:

$$r^* = \beta_1 + \beta_2 \cdot r \tag{2}$$

Plugging in our three distance estimates and their known distances, we get a system of equations which we can easily solve for our parameters β_1 and β_2:

$$d_1^* = \beta_1 + \beta_2 \cdot d_1 \tag{3}$$

$$d_2^* = \beta_1 + \beta_2 \cdot d_2 \tag{4}$$

$$d_3^* = \beta_1 + \beta_2 \cdot d_3 \tag{5}$$

These parameters, along with our linear calibration function, can then be used with any future distance estimate d_t to estimate the true distance d_t^*.

While traditional calibration does successfully give us a mapping from our actual response to our desired response, neither parameter belongs to either the transmitter or the receiver, but rather both belong to the *transmitter/receiver pair*. We will call this *pairwise calibration* since we must repeat this process for every such pair. This is clearly undesirable because the cost of pairwise calibration is order n^2 where n is the number of nodes in the system. Furthermore, we must manage n^2 sets of parameters.

We would prefer to calibrate each transmitter and each receiver individually as we would for, say, a thermometer or magnetometer. That is, we would like to use an order n calibration method. The problem is that, under the traditional calibration framework just described, we need to know both the device response r and the desired response r^*. However, the values of r and r^* in ranging are inherently tied to both the transmitter and receiver.

In Calamari, as we can see, traditional calibration can do pairwise calibration but cannot calibrate individual transmitters or receivers; there is only one response from the system and it cannot separate the effect of the transmitter from the effect of the receiver. We will call this the *separation problem*. In the following sections, we discuss three methods to overcome the separation problem: iterative, mean, and joint calibration.

4 Towards Avoiding the Separation Problem

The separation problem is not specific to Calamari, but to the calibration of all sensor/actuator pairs. We realize the importance of it when we see that *all* calibration is between sensor/actuator pairs. It is usually insignificant when calibrating temperature sensors with room temperature, but might not be when calibrating, for example, magnetometers with a set of magnets.

4.1 Iterative Calibration

The first attempt we see to circumvent the separation problem comes from the ad-hoc localization literature. SpotON [9] uses low-power radio transceivers in an RSSI ranging system very similar to the RSSI component of Calamari. To calibrate, SpotON observes that the separation problem could be avoided if we had at least one calibrated transmitter or receiver. As a solution, it simply *declares* one transmitter T_r to be the reference transmitter and uses it to calibrate all receivers. It then uses one reference receiver R_r to be the reference receiver and calibrate all transmitters.

This procedure effectively *iterates* traditional calibration, so we will call it *iterative calibration*. In both iterations there is only one uncalibrated device in each pair. When a measurement from the pair is in error, the error is attributed to the uncalibrated device, thereby getting past the separation problem.

More precisely, let d_i be the distance between receiver R_i and the reference transmitter and let β_i be that receiver's calibration parameters. For each such receiver, we have a system of equations of the form

$$d_i^* = f(d_i, \beta_i) \tag{6}$$

which we can solve for β_i. We now choose a reference receiver R_r and the distance from it to each transmitter T_j is d_j. For each such transmitter, we have a system of equations of the form

$$d_j^* = f(d_j, \beta_j, \hat{\beta}_r) \tag{7}$$

which we can solve for β_j, where $\hat{\beta}_r$ is the set of parameters for the reference receiver that was learned in the first iteration. Note that the parameters of each receiver are derived only from the distance estimates between it and the reference transmitter, and vice versa for each transmitter.

An interesting part of the SpotON technique is that the receivers are not actually adjusted at all. Instead, each receiver is parameterized in terms of the Seidel and Rappaport RF path loss model. There are two main advantages to using this parameterization. First, the receiver sensitivity cannot be adjusted in hardware anyway. Second, and more importantly, the parameterization can be arbitrarily accurate. In fact, Hightower, et al take advantage of gaussian noise in their data by effectively using least-squares log-linear regression over 100 readings. This technique is expanded

upon in the present work.

There is one serious problem with iterative calibration as applied to SpotON. RSSI is effected by the *difference* in frequency of the transmitter and receiver, as seen in Figure 1. The farther the transmission frequency is from the center frequency of the receiver, the more the incoming signal will be attenuated by the receiver. This means that a receiver's calibration parameters are only valid for a single frequency: that of the transmitter used to calibrate it. Iterative calibration does not actually avoid the separation problem at all; calibration parameters are only known to be valid for one transmitter/receiver pair.

4.2 Mean Calibration

Let us propose one more attempt to avoid the separation problem. Here, we assume that variations in the devices are gaussian distributed. By doing so, we can calibrate all receivers to the *mean* of the transmitters, or vice versa. Let us call this method *mean calibration*. In mean calibration one collects calibration data for each receiver using *all* transmitters as calibrating devices. The parameters learned for the receiver will not be coupled to any particular transmitter and will also minimize the expected error for that receiver in the least-squares sense, assuming that the sample of transmitters used represents the true transmitter population.

More precisely, let $d_{i,j}$ be the distance between receiver R_i and transmitter T_j. For each receiver, we have a system of equations of the form

$$d^*_{i,j} = f(d_{i,j}, \beta_i) \tag{8}$$

which we can solve for β_i. Notice that, unlike iterative calibration, the parameters β_i are not related to any particular transmitter. If we use least squares techniques to solve the system of equations, we are in fact calibrating each receiver to the mean of all transmitters.

Mean calibration trivially avoids the separation problem by not calibrating the transmitters at all. While this system minimizes expected error for the receiver, however, it is unlikely that it minimizes error for the system as a whole since the transmitters have not been calibrated. In the next section, we describe a method that directly minimizes the error of the entire system.

5 Calibration in Calamari

Both methods that we have seen so far are micro-calibration processes. They directly observe each device and build a mapping from r to r^* to directly optimize that device's response. In this section, we will describe a method of macro-calibration that calibrates each device by optimizing the overall system response instead of the individual device responses.

The method has three steps:

1. Parameterize each individual device and model the system response as a whole using these parameters
2. Collect data from the system as a whole
3. Choose the parameters for the individual devices such that the behavior of the entire system is optimized

How does this technique choose parameters for individual devices while only observing the system response? By observing *trends* in the transmitter/receiver pairs, we can attribute errors in the system to the individual nodes that are likely to cause them. In other words, if all distance estimates made with a particular transmitter are slightly high, we can blame that transmitter. By looking at the entire system response at once, we can allocate blame optimally amongst the nodes.

5.1 The Parameterization

We choose parameters for the devices based on our physical understanding of their interactions. We focus here on TOF ranging, although the parameterization is nearly identical for RSSI.

In Calamari, an acoustic pulse of roughly 15ms is transmitted along with a 25ms radio packet. When the microphone receives the acoustic pulse, the phase lock loop (PLL) of the tone detector attempts to identify the signal. When the PLL response and the microphone response are high enough in combination, an interrupt is fired which is time-stamped by the processor. This time stamp is compared with the time stamp of the radio packet. The difference in time is multiplied by the nominal speed of sound to obtain a distance estimate.

Several variations in the hardware strongly effect the TOF readings:

Bias – The times for the sounder and microphone to start oscillating, B_T and B_R, may vary due to variations in the diaphragms

Gain – The volume of the sounder and the sensitivity of the microphone, G_T and G_R, affect the speed with which the PLL detects the signal

Frequency – The difference in sounder frequency and tone detector center frequency, $|F_T - F_R|$, has a near-linear affect on the effective received volume.

Orientation – The relative orientations of the sounder and microphone, O_T and O_R, will affect the volume with which the acoustic tone is received through some non-linear function $f_O(\cdot)$.

We therefore arrive at the following complete model of the system response for a transmitter/receiver pair:

$$\begin{aligned} d^* = B_T + B_R + G_T \cdot d + G_R \cdot d \\ +|F_T - F_R| \cdot d + f_O(O_T, O_R) \cdot d \end{aligned} \tag{9}$$

Where d is the distance estimate and d^* is the true distance between the pair.

5.2 Choosing the Parameters

Assuming we have collected data from our system, we may choose values for the device parameters such that our sensor model above matches the data we collected. To simplify the process, we will remove the non-linear parameters for frequency and orientation and allow both of these factors to be built into the error term. Our simplified but less accurate sensor model becomes:

$$r^* = B_T + B_R + G_T \cdot r + G_R \cdot r \tag{10}$$

Each pair of collected values r and r^* together with this model form an equation with four variables. We collect one sample from each pair in our network, giving us $4n$ variables in $n^2 - n$ equations, where n is the number of nodes. This system of equations will generally over-constrain the values of all device parameters.

More precisely, let $d_{i,j}$ be the distance between receiver R_i and transmitter T_j. For each transmitter and receiver, we have a system of equations of the form

$$d^*_{i,j} = f(d_{i,j}, \beta_i, \beta_j) \tag{11}$$

We cannot solve these equations for either β_i or β_j because there is no unique solution; there is no way to know if error should be attributed to the transmitter or receiver parameters. However, we can use all pairs of nodes to create a system of $n^2 - n$ equations, which we can solve for all $4n$ parameters. Notice that, unlike iterative calibration and mean calibration, we do not calibrate each device by solving a system of equations for that device. Instead, we calibrate the entire system at once by solving a system of equations that relates that system. No parameter of a receiver or transmitter is tied to any device other than that particular receiver or transmitter.

We can use least-squares [2] [15] to solve this system of equations by converting it into matrix notation

$$Ax = b \tag{12}$$

Let x be an array of our device parameters and let each row of A be the coefficients from our model of one data sample b. For example, if our first two distance estimates were $d_{1,2}$ and $d_{1,3}$ coming from transmitter/receiver pairs $(1,2)$ and $(1,3)$, the first two rows of the A and b matrices would be as follows:

$$A = \begin{pmatrix} 1 & 0\cdots & 0 & 1 & 0\cdots & d_{1,2} & 0\cdots & 0 & d_{1,2} & 0\cdots \\ 1 & 0\cdots & 0 & 0 & 1\cdots & d_{1,3} & 0\cdots & 0 & 0 & d_{1,3}\cdots \\ \vdots & & \vdots & & \vdots & \vdots & & \vdots & & \vdots \end{pmatrix}$$

and

$$b = \begin{pmatrix} d^*_{1,2} \\ d^*_{1,3} \\ \vdots \end{pmatrix}$$

where

$$x = \begin{pmatrix} B_{T_1} \\ \vdots \\ B_{T_n} \\ B_{R_1} \\ \vdots \\ B_{R_n} \\ G_{T_1} \\ \vdots \\ G_{T_n} \\ G_{R_1} \\ \vdots \\ G_{R_n} \end{pmatrix}$$

Collecting k distance estimates from our system, we create these matrices and solve for x, where A is a $k \times 4n$ matrix, b is a $k \times 1$ matrix and x is a $4n \times 1$ matrix.

As you can see, this method only uses the estimated distances given by the transmitter/reciever pairs. The actual response of any particular transmitter or receiver was never observed, yet every transmitter and receiver has its own parameters. One big advantage to this method is that any new nodes added to the network will not have to be pairwise calibrated with each existing node but only with a sample of transmitters and receivers.

In the next two sections we empirically evaluate this method and compare it with the methods previously described.

6 Experimental Setup

In this section we describe the exact hardware and methodology used to collect data and evaluate these calibration methods.

Calamari is built on the MICA sensor platform [10]. The MICA mote is essentially an Atmel 103 microcontroller in conjunction with a RFM TR1000 radio transceiver and is about the size of two

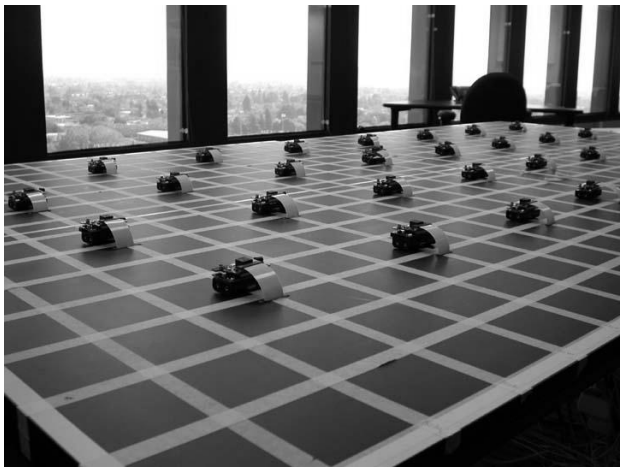

Figure 2: **Experimental Testbed** *consists of a 8 x 4 grid of 32 nodes at 30cm spacing*

AA batteries, as seen in Figure 3. Connected to each mote is a MICA sensor board [18] which, among other things, contains a Sirius PS14T40A 4.3KHz sounder and a Panasonic WM-62A microphone, whose band-passed output is wired to a National Semiconductor LMC567CM tone detector whose center frequency is set to about 4.5KHz.

In our experiments we use 32 nodes in a 30cm x 30cm grid. The motes are situated atop a large table in an 8 x 4 formation spanning a total area of 210cm x 90cm, as seen in Figure 2. All motes are in the same orientation. Both the sounder and microphone point directly upwards. The sounder is located in the center of the board while the microphone is in one corner, as seen in Figure 3.

Although this experimental setup does not test all combinations of paired distances and relative orientations, it does provide a nice sampling of that space. A total of 24 distances ranging from 30cm to 210cm are measured using this setup. While all motes are in the same orientation on the table, they are not all in the same orientation relative to each other. A total of 992 transmitter/receiver pairs are used, although each at only one distance and orientation. The acoustic transmission frequencies are distributed evenly from 4.2KHz to 4.5KHz while the tone detector center frequencies are distributed between 4.4KHz and 4.6KHz.

To generate the data, each transmitter sent several localization beacons allowing all other nodes to make a single distance estimate. Each node therefore generated up to 31 distance estimates, all

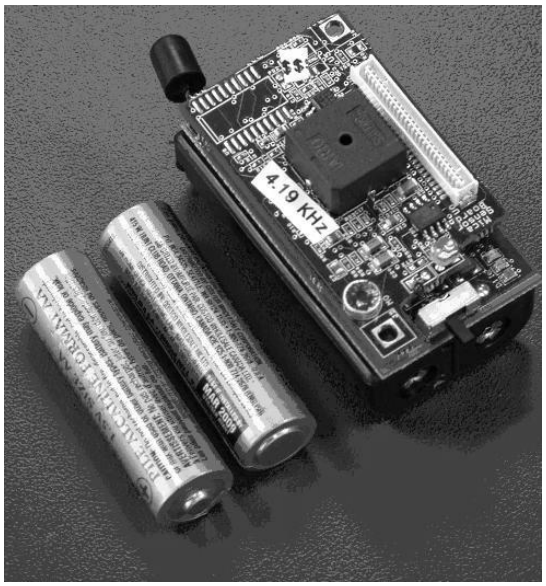

Figure 3: Mica Sensorboard Mounted on a Mote

of which are plotted against their corresponding true distances in Figure 4. It is visually apparent that the uncalibrated distance estimates are very poor; the average error is 74.6%.

Note that all distance estimates are greater than or equal to the true distance. This is because a signal will never be received faster than the speed of sound although it may be detected well after it initially arrives.

7 Evaluation

For comparative purposes, we use linear regression to map the distance estimates to their true distances in what we call *uniform calibration* since it characterizes all transmitters and receivers in a uniform way with a single set of parameters, not accounting for individual variations among the devices. The average error is 21% with uniform calibration, as seen in Figure 5.

In this section we evaluate and compare the three methods of parameter estimation: iterative calibration, mean calibration, and joint calibration. To make the evaluations comparable, we use the same parameters and calibration function in all cases and evaluate them using the same data.

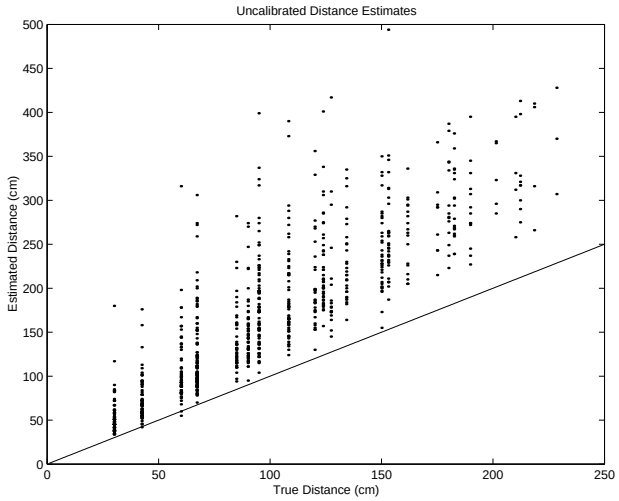

Figure 4: **Uncalibrated TOF readings** *are always greater than the true distance and are highly erroneous due to transmit- and receive-delays.*

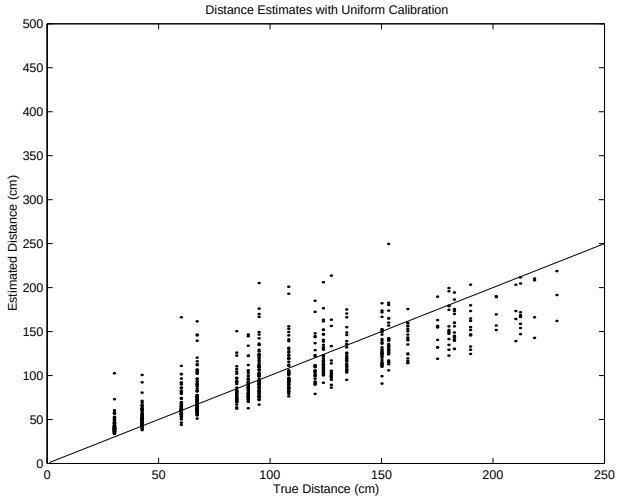

Figure 5: **Uniform Calibration** *is essentially linear regression. It accounts for the fact that transmit- and receive-delay exists, but does not account for the values of delay associated with each transceiver.*

Recall that our calibration function is

$$r^* = B_T + B_R + G_T \cdot r + G_R \cdot r \tag{13}$$

By convention, we will assume an uncalibrated node has parameters $B = 0$, $G = \frac{1}{2}$. This gives us the reasonable model of two uncalibrated nodes:

$$r^* = 0 + 0 + \frac{1}{2} \cdot r + \frac{1}{2} \cdot r \tag{14}$$

7.1 Iterative Calibration

For iterative calibration we chose an arbitrary node to be the reference transmitter, which we call transmitter A, that we will use to calibrate all receivers. We assign it parameters $B_T = 0$ and $G_T = \frac{1}{2}$. Then, for each receiver R, we generate the matrices below with all distance estimates $d_1, \ldots, d_k$ collected from transmitter A and receiver R. We then solve for the receiver parameters B_R and G_R using least-squares.

$$x = \begin{pmatrix} 0 \\ \frac{1}{2} \\ B_R \\ G_R \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & d_1 & 1 & d_1 \\ 1 & d_2 & 1 & d_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & d_k & 1 & d_k \end{pmatrix}$$

$$b = \begin{pmatrix} d_1^* \\ d_2^* \\ \vdots \\ d_k^* \end{pmatrix}$$

We then chose an arbitrary reference receiver which we call receiver B and use it to calibrate all transmitters. This iteration is identical to the first except that we solve instead for B_T and G_T

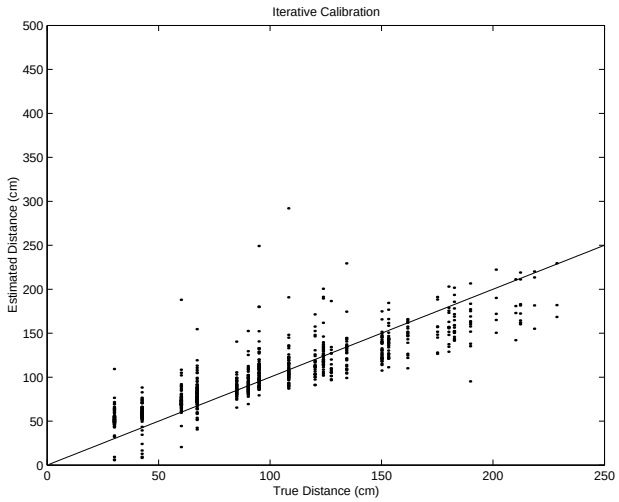

Figure 6: **Iterative Calibration** *attempts to address differences in transceivers, but suffers from error propagation.*

using the x matrix below, where $\hat{B_B}$ and $\hat{G_B}$ are values of the parameters learned for receiver B in the first iteration.

$$x = \begin{pmatrix} B_T \\ G_T \\ \hat{B_B} \\ \hat{G_B} \end{pmatrix}$$

Now that all calibration parameters have been set, we use these parameters along with our calibration function (13) to calibrate all of our readings. The calibrated readings are shown in Figure 6, where the average error was reduced from 21% in uniform calibration to 19.7% in iterative calibration. Notice in particular that, although the overall error was reduced, many estimates were not affected and that several others actually got worse.

7.2 Mean Calibration

Mean calibration is identical to the first iteration in iterative calibration except that we use *all* transmitters instead of just one *standard* transmitter. All transmitters are to remain uncalibrated, so we assign them parameters $B_T = 0$ and $G_T = \frac{1}{2}$. For each receiver R, we generate the same matrices as the first iteration of iterative calibration above but use the distance estimates $d_1, \ldots, d_k$

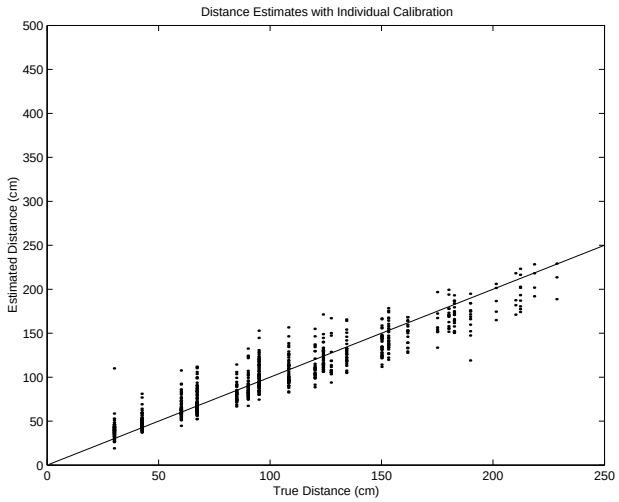

Figure 7: **Mean Calibration** *calibrates each receiver to the mean of all transmitters or each transmitter to the mean of all receivers. This approach works better than iterative calibration but does not allow one to calibrate both receivers and transmitters.*

collected from *all* transmitters and receiver R. We then solve for the receiver parameters B_R and G_R using least-squares.

Having calibrated all receivers, we plot the calibrated distance estimates in Figure 7, where the average error is reduced from 19.7% in iterative calibration to 16.0% in mean calibration.

7.3 Joint Calibration

Using the method of joint calibration as described in section 5, we obtain the results in Figure 8, where the average error has been reduced from 16.0% in mean calibration to 10.1% in joint calibration.

8 Parameter Management

Calamari uses what we call a *piggy-backing* protocol in that localization beacons are fundamentally tied to radio communication packets. When a node sends a radio packet, it optionally also sends an acoustic pulse. There are three advantages to this system. First, all radio messages are already timestamped on arrival, giving us a foundation for TOF readings at no additional cost. Second,

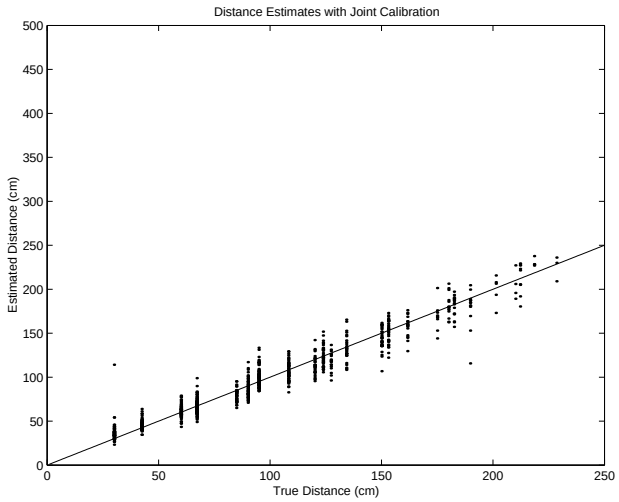

Figure 8: **Joint Calibration** *assigns parameters to each device such that overall system performance is optimized.*

since the RF range is larger than the acoustic range, we don't have beacon collision problems. If the beacon is accompanied by a radio packet, we can identify the source; if it is not, there must have been a packet collision and therefore potentially a beacon collision, so we ignore all acoustic information. The third advantage to the piggy-backing protocol is that we do not have to worry about scheduling localization beacons. We essentially hand the problem off to the wireless networking community; as they increase the throughput of wireless networks, we automatically see more efficient beacon scheduling. Notice that the corresponding disadvantage is that a communication-starved node is also location-starved.

Another benefit of using this piggy-backing protocol is efficient management of our calibration parameters. Each device maintains its own calibration parameters and sends them in the payload of each localization beacon along with its current position estimate. Storing the parameters in a distributed manner allows the network to change and grow dynamically with no overhead cost for management or infrastructure. Furthermore, each node can autonomously adjust its own calibration parameters over time, perhaps using techniques such as autocalibration as described in the next section.

9 Autocalibration

The reason why joint calibration is so much more successful than iterative and mean calibration is that it used a more general parameter estimation technique where regression in the strict sense failed. This method can in fact be extended to arbitrarily complex parameter estimation. One such example is proposed in this section, where we frame calibration as a constrained optimization problem.

In the previous sections, we *manually* calibrated the devices in the sense that we had a carefully controlled environment; we knew the true distance between each node. Let us define autocalibration as calibration in an *un*controlled environment; we do not know the true distance between each node.

Many sensor systems have incorporated an automatic calibration mechanism, perhaps for the motors of a robotic arm [14] or for an electronic compass. However, autocalibration invariably requires specialized hardware, such as the use of gyroscopes [11] or cameras.

Instead of using extra hardware in Calamari, we will use *a priori* information about localization to try to *infer* our device parameters.

The first peice of *a priori* information is that in Calamari every transmitter/receiver pair is also a receiver/transmitter pair. In other words, there are two distance estimates for each pair of motes. One way to select parameters is to select those that maximize the consistency of the responses between these symmetric pairs. This is to say, for nodes i and j, we want:

$$d^*_{ij} = d^*_{ji} \tag{15}$$

where $d^*_{ij} = B_{T_i} + B_{R_j} + G_{T_i} \cdot d_{ij} + G_{R_j} \cdot d_{ij}$,

The second peice of *a priori* information is that all distance estimates must satisfy the triangle inequality. In other words, for any three nodes i, j, and k that are all connected:

$$d^*_{ij} + d^*_{jk} - d^*_{ik} \geq 0 \tag{16}$$

Together, these two hooks into the problem give us a constrained optimization problem: maximize consistency while satisfying the triangle inequality. To prevent the degenerate case where all

parameters equal zero, we must add two more terms to the objective function: $\sum_i (G_{T_i} - 1)^2$ and $\sum_i (G_{R_i} - 1)^2$. A quadratic programming formulation arises:

Minimixe:

$$\sum_{i<j} (d^*_{ij} - dji^*)^2 + \sum_i (G_{T_i} - 1)^2 + \sum_i (G_{R_i} - 1)^2$$

Subject to

$$d^*_{ij} + d^*_{jk} - d^*_{ik} \geq 0 \qquad \forall\, triangle_{i,j,k}$$

Notice that if we actually knew some distance $\hat{d}_{ij}$, the known distance could easily replace the estimate d^*_{ij} in the quadratic program above. This would allow us to easily take advantage of anchor nodes or pre-calibrated nodes.

Furthermore, if we knew all distances this quadratic program would reduce to minimizing the squared error between all distance estimates and their corresponding true distances. This would effectively reduce the program to the least-squares approach used in joint calibration.

It is worthwhile to note that distance-based constraints were previously known to be useful in the localization literature [7]. However, that paper differs from this work in that it maps out the convex set of feasible locations for localization purposes whereas the constraints above map out the convex set of feasible distances for calibration purposes.

10 Future Work

While joint calibration is a fine solution for manual calibration, work in autocalibration is clearly experimental. Calibration as constrained optimization will probably only work under certain network geometries where the device parameters are actually over-constrained. There are also complexity issues with this technique since each clique of size n in the network has $\binom{n}{3} * 2^3$ constraints based on the triangle inequality alone. Further, numerical issues may prevent the program from being solvable at all; a quadratic program is convex if and only if the coefficient matrix is positive semidefinite [5] [3]. Current work includes solving these issues as well as arriving at a linear approximation to the quadratic program.

Frequency and orientation affect our distance estimates enough that another natural extension of this work is to use more sophisticated machine learning techniques that can handle non-linear parameters. For example, we could build a belief network using empirical data that describes the behavior of the system over the entire parameter space. Then, we could us any probabilistic reasoning method such as MCMC to infer the bias, volume, frequency, and orientation of all the devices.

11 Generalizing Calibration as Parameter Estimation

Calibration as parameter estimation can be generalized to other sensor network applications. In fact, we have already seen it used in one: time synchronization for sensor networks, which we view here as a method of calibrating *time sensors*. The method proposed by Elson and Estrin [6] compares clock values of different clocks upon the arrival of an RF reference broadcast. The broadcast arrives at all clocks simultaneously, allowing us to compare values from the same time instance. This method, in fact, generalizes to the calibration of any sensors in the presence of a uniform yet constantly changing stimulus. For example, reference broadcasts would be useful for the relative calibration of temperature sensors in the presence of perfectly uniform, constantly changing temperature.

In this work, the clocks are not actually adjusted at all but are parameterized in terms of skew and offset and a calibration function is used to translate between the values of different clocks. Similar to Hightower, et al [9], Elson and Estrin take advantage of gaussian noise in their data by effectively using least-squares linear regression over multiple readings. This technique was expanded upon in joint calibration and is generally useful when a higher degree of calibration is desired than can be realized by the device.

Joint calibration might be useful to general applications needing *relative* calibration. Iterative calibration is currently the *de facto* standard for relative calibration, which is the method of choosing one sensor as the *standard* and calibrating all other sensors against it. This method may cause problems, for example, with light sensors that respond to each light source differently, perhaps due to their different frequencies. It also may be problematic when calibrating magnetometers to specific magnets whose orientations might be slightly askew. Using joint calibration will prevent the

specific device used for calibration from affecting the calibration parameters of the light sensor or magnetometer.

Autocalibration as described in this paper may be most useful to other sensor network applications. This is especially true because autocalibration is often a requirement when using sensors with drift or when one needs to calibrate for an unknown environment. For example, the drift of a temperature sensor might be controlled by constraining its response relative to other nearby temperature sensors. The environment of a field of magnetometers could be automatically calibrated for if all devices were getting extra high or low readings.

Finally, constraint-based methods could serve as a mathematical framework for finding faulty or malicous sensors or anomolies in the sensor field. In such cases, the set of feasible solutions to the constrained optimization problem would be the empty set. For example, if estimating the position of a magnet in the presence of ambient magnetic anomolies, there would be no position of the magnet that would give rise to such a magnetic field. Similarly, there may be no set of calibration parameters that would cause a temperature sensor to give a certain pattern of erroneous readings, so the sensor must be either faulty or malicious. Finally, in the case of localization, the triangle inequality can be used to identify multi-path problems. The multi-path readings from a particular transmitter would violate the triangle inequalities used in our constrained optimization program, and there would be not set of calibration parameters that would explain why that transmitter behaves well with some nodes and poorly with others.

12 Summary and Conclusions

Micro-calibration is inadequate for the calibration of hundreds or thousands of general-purpose devices, perhaps with no calibration interfaces, that may be in an uncontrolled or unobservable environment or may even be unobservable themselves. Sensor networks demand a method of macro-calibration.

Calibration as generalized parameter estimation may be taking one step in that direction. It gives us a calibration interface to devices that don't already have one and allows us to calibrate

general-purpose devices for specialized tasks, such as ranging.

Another advantage of calibrating in software is that we can leverage the redundancy and distributed computational power of sensor networks to have the network calibrate itself. This makes it feasible to calibrate even very large networks.

Finally, the most important benefit of calibration as parameter estimation is that we can use arbitrarily sophisticated parameter estimation techniques to take advantage of redundancy and *a priori* knowledge about our application to calibrate in otherwise impossible conditions. These include unknown or unobservable environments, or even simply separately calibrating a sensor and an actuator system where neither device is already calibrated.

Acknowledgments

Many thanks to Rob Szewczyk for help with the mathematical analysis and to Alec Woo and Jason Hill for assistance with the hardware used in *Calamari*. This work was supported by the Intel Research Laboratory at Berkeley, by DARPA grant F33615-01-C-1895 (Network Embedded Systems Technology "NEST") and by the NDSEG Fellowship Program. This work was done in conjunction with the University of California Center for Information Technology Research in the Interest of Society (CITRIS).

References

[1] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, March 2000.

[2] R. Barrett, M. Berry, and et al. T. F. Chan. Templates for the solution of linear systems: Building blocks for iterative methods. In *SIAM*, Philadelphia, 1994.

[3] T.F. Coleman and Y. Li. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.

[4] copradar. Traffic radar handbook. http://copradar.com.

[5] crossbow. Lp series accelerometer specifications. http://www.xbow.com/Products/Product_pdf_files/LP%20Accel.pdf.

[6] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), May 2002.

[7] L. Doherty et al. Convex position estimation in wireless sensor networks. In *Infocom*, Los Alamitos, Calif., 2001. IEEE CS Press.

[8] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *Mobile Computing and Networking*, pages 59–68, 1999.

[9] Jeffrey Hightower, Chris Vakili, Gaetano Borriello, and Roy Want. Design and calibration of the spoton ad-hoc location sensing system. August 2001.

[10] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *ACM ASPLOS IX*, November 2000.

[11] Bruce Hoff and Ronald Azuma. Autocalibration of an electronic compass in an outdoor augmented reality system. In *IEEE and ACM International Symposium on Augmented Reality*, pages 159–164, Munich, Germany, October 2000.

[12] B. Hofmann-Wellenho, H. Lichtenegger, and J.Collins. *Global Positioning System: Theory and Practice*. Springer Verlag, fourth edition, 1997.

[13] ISKS. Isks ocean monitoring system. http://www.seakeepers.com/installation_manual.pdf.

[14] D. Ma and J. M. Hollerbach. Identifying mass parameters for gravity compensation and automatic torque sensor calibration. *IEEE Intl Conf Robotics and Automation*, pages 661–666, April 1996.

[15] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Soft.*, 8:43–71, 1982.

[16] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000.

[17] Kamin Whitehouse. The design of calamari: an ad-hoc localization system for sensor networks. Master's thesis, University of California at Berkeley, 2002.

[18] Alec Woo. Mica sensor board. http://today.cs.berkeley.edu/tos/hardware/hardware.html.

[19] J. Zhang Y. von Collani and A. Knoll. A generic learning approach to multisensor based control. The International Conference on Multisensor Fusion and Integration of Intelligent Systems, Baden-Baden, 2001.