The Design and Implementation of a Self-Calibrating Distributed Acoustic Sensing Platform[*]

Lewis Girod
Computer Science and AI Laboratory
Massachusetts Institute of Technology
Cambridge, MA, 02139 USA
girod@nms.csail.mit.edu

Martin Lukac   Vlad Trifa   Deborah Estrin
Center for Embedded Networked Sensing
University of California, Los Angeles
Los Angeles, CA, 90095 USA
{mlukac,destrin}@cs.ucla.edu,
vlad.trifa@ieee.org

Abstract

We present the design, implementation, and evaluation of the Acoustic Embedded Networked Sensing Box (ENSBox), a platform for prototyping rapid-deployable distributed acoustic sensing systems, particularly distributed source localization. Each ENSBox integrates an ARM processor running Linux and supports key facilities required for source localization: a sensor array, wireless network services, time synchronization, and precise self-calibration of array position and orientation. The ENSBox's integrated, high precision self-calibration facility sets it apart from other platforms. This self-calibration is precise enough to support acoustic source localization applications in complex, realistic environments: e.g., 5 cm average 2D position error and 1.5 degree average orientation error over a partially obstructed 80x50 m outdoor area. Further, our integration of array orientation into the position estimation algorithm is a novel extension of traditional multilateration techniques. We present the result of several different test deployments, measuring the performance of the system in urban settings, as well as forested, hilly environments with obstructing foliage and 20–30 m distances between neighboring nodes.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Signal processing systems*

General Terms

Algorithms, Design, Experimentation, Measurement

Keywords

Self-Localization, Distributed Acoustic Sensing

[*]This work was made possible with support from the NSF Cooperative Agreement CCR-0120778, and the UC MICRO program (grant 01-031) with matching funds from Intel.

1 Introduction

Distributed acoustic sensing has many applications in scientific, military, and commercial applications, including population measurement projects tracking the calls of birds [37] and wolves, military systems tracking vehicle [39] and personnel [2] movements, and commercial systems in support of smart spaces. Acoustic source localization can also provide an inexpensive and easily integrated solution to the more general sensor node localization problem, by using a source-localizing infrastructure to detect and locate small, inexpensive nodes that emit a characteristic calibration signal. However, despite the overall interest in these problems, and despite significant progress in related areas such as source localization theory and sensor network systems, progress toward developing and deploying these applications has been greatly slowed by the absence of an integrated platform suitable for prototype acoustic source localization systems. While prior acoustic sensing projects have developed systems to support specific applications, those systems have either been too heavily optimized and too application-specific to support rapid prototyping, or else have lacked a feature set appropriate to acoustic source localization.

Figure 1 shows a typical distributed acoustic source localization application. Several sensor array nodes are located at points surrounding an event of interest. When an event of interest occurs, the system should output an estimate of the most likely location of the event, while other sources of acoustic energy should be filtered out as noise.

In a typical implementation, each node runs a lightweight streaming detection algorithm on a single acoustic channel to search for events that match a certain signature. When an event matches, full array data is retrieved and more sophisticated processing is scheduled to compute a bearing estimate and to enhance the signal. These estimates and other data are then correlated among the reporting sensors to estimate a probability distribution of the most likely locations of the target source [37].

In this paper we present the Acoustic Embedded Networked Sensing Box (ENSBox), a platform for prototyping rapidly-deployable distributed acoustic sensing systems, particularly distributed source localization. Each ENSBox integrates a developer-friendly ARM/Linux environment with key facilities required for source localization: a sensor array,

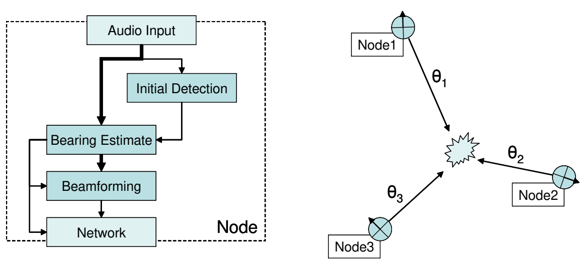

Figure 1. Diagram of a typical source detection and localization application. The left shows typical processing at the sensor, while the right shows cross-beam localization. Note that the node orientation must be known in order to interpret bearing estimates relative to the array.

network services, time synchronization, and self-calibration of array position and orientation. Self-calibration is especially important to source localization applications, because error in the assumed orientation of a node directly offsets the bearing estimates, resulting in a localization error that scales with the range to the target. Similarly, error in the position of a node can result in a direct translation in the localization result. The ENSBox's integrated, high precision self-calibration facility eliminates manual survey of array positions and orientations, capturing the imagination of a number of scientists interested in bio-acoustics who would otherwise need to develop their own platform. Further, our integration of array orientation into the position estimation algorithm is a novel extension of traditional multilateration techniques.

To ensure our system's viability for typical source localization applications, we performed outdoor test deployments in a variety of environments. In these tests we achieved very high accuracy estimates of array position and orientation, results an order of magnitude better than prior acoustic work [20]: 5 cm average 2D position error and 1.5 degree orientation error over a partially obstructed 80x50 m area. Our results are comparable to recent RF results [23], but since our application itself involves acoustic sensing, using acoustic signals to calibrate the arrays is more direct.

We attribute much of this success to our system architecture, which enabled us to build a simpler, less optimized system that in turn could support more sophisticated and effective signal processing algorithms. As a final indicator of the success of our platform, several other groups have already begun using it in source localization research.

In the following sections, we present the design, implementation, and evaluation of the Acoustic ENSBox system. §2 discusses related work in sensing platforms and self-localizing systems. §3 describes the Acoustic ENSBox platform. §4 describes our high-accuracy self-calibration system. §5 defines performance metrics and presents a thorough evaluation of our self-calibration system in several realistic outdoor environments. Finally, §6 discusses these results in the context of related work.

2 Related Work

The related work for this project falls into two categories: platforms designed to support acoustic sensing, and self-localization systems. In this section we discuss the related work in both of these areas.

2.1 Acoustic Sensing Platforms

Support for acoustic sensing can be found in a variety of off-the-shelf solutions and research projects in the sensor network field. These solutions vary greatly in terms of the facilities they provide as well as how amenable they are to prototyping.

The most off-the-shelf solutions are PC hardware or laptops with multi-channel sound cards, or even appliances such as wireless networked web-cams. While they are readily available and easiest for an end-user to pick up and use, they tend to be more difficult to use in a distributed context. Solutions that stream all of the data to a central point typically don't scale, and implementing tight time synchronization across nodes is quite difficult because off-the-shelf audio products are rarely designed to provide sample-accurate time stamps.

A number of research projects in the sensor network field have developed platforms in the course of their work. One of the earliest general-purpose sensing platforms was the WINS NG platform developed by Sensoria Corp. for the DARPA SensIT program [26]. This platform supported multi-channel sensing, multihop wireless networking and a Linux OS with few resource constraints. However, it did not support tight time synchronization; most of the results using this platform were synchronized off-line using a starting pistol as a marker. Time synchronization features were added in a follow-on project developed for the DARPA SHM program [25], but this version was a closed system, and was not made available to a larger community.

A number of projects have developed acoustic systems based on the Berkeley Mote [16] and the MoteIV Telos product.[1] The Line in the Sand [1] and Extreme Scaling [2] demos of the DARPA NEST program demonstrated acoustic sensing platforms based on the XSM platform. These projects successfully used microphones, magnetometers, and IR to track soldiers walking through a large sensor field, but limited memory, CPU and communications capacity ruled out complex signal processing algorithms.

The Countersniper system developed at ISIS [22] is an example of a platform that provides tight time synchronization and high-speed sampling. However, because both hardware and software are heavily optimized for the countersniper application, it is difficult to use as a prototyping platform.

The VanGo [15] project uses a Telos as a platform for processing audio, with the property that processing elements can be dynamically shuffled between the Telos and the PC. However, it does not provide time synchronization and the processing limitations of the Telos make sophisticated implementations difficult to achieve without early optimization.

2.2 Self-Localization Systems

Self-localization systems have been an active area of research for many years [38] [28] [33]. Due to space constraints we will mainly discuss other work designed for outdoor use. The most similar projects are three audible acoustic ranging systems described by Sallai [31], Kwon [20], and Kushwaha [18]. These systems vary in performance, but also vary greatly in their available RAM and computational

[1] See `http://www.moteiv.com`.

power. The systems described by Sallai and Kwon are both implemented using a standard Mica2, and in the case of Sallai, a standard sensor board. The system reported by Kushwaha is based on a Mica2 with an attached custom 50 MHz DSP processor and an external speaker. While these systems have the advantage of running on much simpler, much lower-power hardware, we will see in §6 that the Acoustic ENSBox leverages its greater computational resources to achieve higher accuracy and longer ranges in more complex environments.

Recent work in radio-interferometric localization [23] has been implemented using the Mica2 platform, and shows much promise. In open-space testing with minimum multipath interference, this work demonstrates comparable ranging and localization accuracy to our acoustic system. However, this technique may be susceptible to errors from multipath interference, and no results from more complex, cluttered environments have been published.

The Cricket compass [29] is an ultrasound-based bearing estimator intended for pervasive computing applications. The bearing estimation aspect of our system is similar, although our techniques yield higher accuracy. We will make a more detailed comparison in §6.

3 Platform Overview

The Acoustic ENSBox system provides a platform for developing deployable prototypes of distributed acoustic source localization and sensing applications. We achieve this by providing the necessary hardware and software facilities in a platform that has sufficient resources to deploy systems without extensive optimization. These facilities include a Linux operating environment, a sensor array on each node with a data sampling service, sub-sample time synchronization across nodes, communication services, and a self-calibration service that automatically determines the positions and orientations of the deployed arrays. This platform is described in more detail in [10], and previous versions are described in [36] [11].

3.1 Hardware

The Acoustic ENSBox is based on the Sensoria Slauson board, a single board computer based on the 400 MHz Intel PXA255, with 64MB RAM, an on-board 32MB flash, and no FPU. The CPU board includes an SD-card slot for additional storage, and a dual slot PCMCIA interface, which hosts an 802.11 wireless interface and a Digigram VXPocket440 four-channel sampling card. The node runs the Linux 2.6.10 kernel, with minor modifications to the kernel and to the Digigram firmware required to support accurate timestamping of sensor data. Application and other user-space software is written within the Emstar [12] software environment.

Each node hosts a 4-channel microphone array, geometrically arranged as shown in Figure 2. The microphones are condenser modules (RTI 1207A) coupled with a custom preamplifier board. They are mounted securely in a plastic and aluminum chassis that is readily mounted atop a tripod or stake.

The node can be powered from an internal Li+ battery or from an external source such as an adapter, an external battery, or a solar panel. The system will run continuously

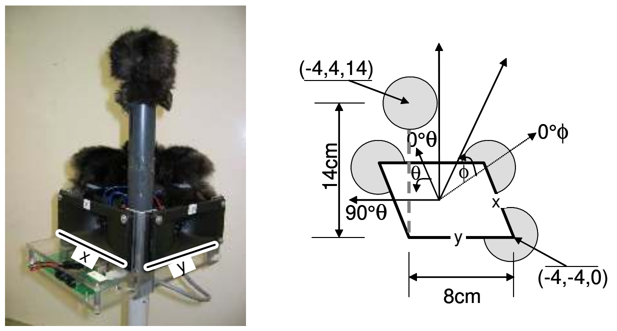

Figure 2. Photograph of an acoustic array, and a diagram of the *local* coordinate system defined relative to the array geometry. The microphones are laid out in an 8 cm square, with one raised 14 cm above the plane. The origin is at the center of the plane. The azimuth angle θ defines the positive X-axis as 0 degrees, increasing counterclockwise in the plane, and the zenith angle φ defines the ray parallel to the plane as 0 degrees.

for 24 hours on a single 12V 7.2 AH (86WH) gel cell. This compares favorably with laptop run times, which typically run for 3 hours on a 50WH battery. Significantly longer lifetimes should be possible with duty cycling, but we leave this to future work.

3.2 Software Services

While hardware integration is an unavoidable part of building a platform, the key advantages of the Acoustic ENSBox platform lie in the software and API stack that we have designed to support our target applications. These include a time synchronized sampling API, networking primitives to support in-network collaboration, and a location and orientation estimation service.

3.2.1 Data Sampling

Many sensing applications run an on-line detection process that triggers more sophisticated *post-facto processing* on the portions of the stream most likely to contain events. This technique is used locally to reduce false positive rates, and to trigger remote nodes to perform collaborative processing. Post-facto processing yields an intuitive solution to this data-flow structure, because the designer can abstract away non-deterministic system delays, e.g. network latency.

To support this model, the data sampling interface defines a persistent and continuous sample "clock" for the sensor inputs, and preserves recent historical data in a ring buffer. The interface enables access to historical data by index range as well as streaming access to incoming data. It also integrates with a time synchronization system that maintains clock conversion parameters among a set of local sensor and remote node clocks. This implementation follows from earlier versions described in [36] [11], and is described in more detail in §3.2.2.

For example, consider the detection system shown in Figure 1. In this case, the initial detection algorithm processes one channel of acoustic data, and identifies features of interest. Upon detection, it passes the index range to another module that retrieves the corresponding data from all four channels, computes a bearing estimate, and enhances the signal using beamforming. The internal buffer enables post-facto retrieval of the four channel dataset, after the initial detection has triggered.

This model is even more compelling in the case of triggering across nodes, where network latencies are involved. After detection and enhancement at node 1, the time and bearing of the detection are sent to node 2. Using the time conversion API, node 2 can convert the time in the message to a range of data from its own sensor clock, and retrieve that data for further analysis.

3.2.2 Multihop Time Conversion

Collaboration across nodes in a distributed system requires facilities for reconciling the timing of events recorded at different nodes. The precision required varies for different applications: those involving measurement of time of flight or time difference of arrivals often require precisions on the order of microseconds [32], while for long-term recording of event times and for many system mechanisms, millisecond or second resolution is sufficient. However, despite these variations, some form of time synchronization is one of the most common and critical requirements in embedded sensing. Acoustic ENSBox supports time synchronization with precision on the order of 10 microseconds over multiple RF hops, satisfying the requirements of the self-calibration service.

The Acoustic ENSBox supports an integrated suite of time conversion facilities, briefly introduced in §3.2.1. This *conversion* approach, first proposed in [9] and later formalized in [19], differs subtly from traditional approaches to time *synchronization*. Whereas *synchronization* methods discipline the clocks to control their rate relative to a standard, *conversion* methods allow the clocks to run independently and instead produce or maintain conversion parameters that can *convert* a point in one timebase to another on demand. This approach is advantageous from a systems and integration perspective, because disciplining an oscillator without introducing artifacts generally requires specialized hardware support.

The time conversion API serves as a broker between services and clients: services that manage a resource containing a clock publish time relations, and clients request conversions. The Acoustic ENSBox platform presents applications with three pre-defined clocks: the node's local CPU clock (i.e. the output of gettimeofday()), the local sensor clock, and global time. In addition, the system maintains conversion metrics to the CPU clocks of one-hop neighbors over 802.11, using Reference Broadcast Synchronization (RBS) [8]. RBS is unique in that it can yield synchronization on the order of microseconds using standard 802.11 hardware; solutions such as NTP [27] typically yield 100 microsecond precision over 802.11 [8]. Attempts to synchronize based on microsecond-granularity timestamps from the 802.11 MAC layer have also failed, because the MAC clocks do not maintain linearity for more than 10s of seconds [10].

Acoustic ENSBox supports two methods of multihop time conversion. The first method is to place the timestamp of interest in a network packet, and convert that timestamp to "local time" on every hop through the network. This method is supported by the flood routing service, for certain known packet types. The advantage of this approach is that it does not require any coordination to determine a global timebase, since it simply converts from the timebase of the source to that of the destination, along the path between the two nodes. The disadvantage is that while it provides accurate conversions it does not provide a good way of storing timestamps for future interpretation, because over long periods of time the clocks will not behave linearly.

The second method is to use the global time service. The global time service uses the first method to push out a message containing a fixed timestamp from "global time", along with the corresponding timestamp in local time. As this message propagates, the local timestamp is converted after each hop into the timebase of the receiver, thus providing each node with an observation of "global time" in terms of its local clock. Once several of these observations are known, a linear conversion relation can be derived using least squares. Thus, if some subset of the nodes have access to time from a GPS unit, they can "broadcast" global time into the network. Applications can then simplify their protocols and leverage the superior frequency stability of GPS by reporting and recording events in terms of global time.

The support for multihop time synchronization provided by the Acoustic ENSBox fits into the API framework proposed in Elapsed Time on Arrival [19]. The ENSBox sampling service provides what ETA terms a Data Series Timestamping API. The ENSBox hop-by-hop conversion mechanism provides an Event Timestamping API, and is mechanically similar to the "RITS" algorithm. The ENSBox global time service provides a Virtual Global Time API, and is mechanically similar to the "RATS" algorithm. However, in the low-level implementation, RBS is used as the synchronization primitive, rather than ETA. Unlike RBS, an ETA implementation over 802.11 radios would require firmware modifications that exposed precise timing of message arrival and transmission, as well as the ability to add timing data to a packet directly before transmission. While ETA should in principle always outperform RBS, ETA's message timing requirements are impossible to implement for systems in which ETA is not explicitly supported and the implementation of the radio is closed.

3.2.3 Communication

Distributed sensing applications inevitably rely on network facilities. While solutions such as Roofnet [5] can provide end-to-end IP routing, many applications can benefit from other network level abstractions. The Acoustic ENSBox platform includes support for two primitives developed to support system diagnostics and self-calibration.

The first primitive is a unreliable hop-scoped flooding service with integrated hop-by-hop timestamp conversion. This mechanism provides a simple way for an application to propagate an event notification with a precise timestamp. While the flood is not guaranteed reliable, flooding generally yields high reliability with low latency.

The second primitive is a *reliable* publish-subscribe mechanism for typed key-value data [14]. Using this layer, applications publish tables of data that are subsequently received reliably at all nodes within a defined hop radius. Because the implementation is based on publishing a sequenced log of updates, it can publish small updates to the previous state efficiently.

The introduction of a reliable publication layer simplifies the implementation of many aspects of the platform. Components on each node use this layer to report hardware faults and to publish range and bearing estimates to other nodes. Similarly, the centralized position estimation algorithm uses it to publish the results of position estimation. By designing the system based on scoped broadcast semantics, we avoid the added complexity involved with explicit coordination points.

3.2.4 *Self-Calibration*

Support for source localization is one of the primary goals of the Acoustic ENSBox platform. In these types of applications, bearing estimates to the source and signal arrival times are measured at several locations; these estimates are then combined to yield an estimate for the location of the source. However, meaningful interpretation and combination of these observations requires precise knowledge of the positions and orientations of the nodes measuring bearing and arrival time.

To address this, the Acoustic ENSBox includes software that implements a self-calibration service that can determine the 3D location and orientation of the sensor arrays in the system. As shown in Table 1, this service achieves very high precision in outdoor tests, with average 2D position error of 5 cm in an outdoor 80x50 m daytime urban environment, and with average error in orientation estimates of 1.5 degrees. We will discuss the design and implementation of this service in detail in §4, and present the results of performance testing in §5. The design of this service is particularly relevant because it exercises the same properties of the platform as are required for many of our target applications. In fact, the data sampling, time synchronization, and network facilities described in the preceding sections were designed to support this localization service.

Trial		2D (cm)		3D (cm)		Orientation	
		Avg.	Med.	Avg.	Med.	Avg.	Med.
CY1	1	6.0	6.6	57.3	62.1	1.0	0.9
	2	5.1	5.1	38.1	35.4	0.8	0.7
CY2	1	4.2	3.7	36.4	22.0	1.6	1.3
	2	4.4	4.4	49.6	44.4	1.9	1.5
JR	1	9.1	10.0	41.9	36.3	2.7	1.7
	2	7.2	6.5	32.5	32.5	3.2	2.4
	3	8.2	7.7	36.2	20.5	3.8	2.4
	4	9.7	11.3	26.0	24.3	1.8	1.7
	5	11.1	12.2	30.6	29.8	1.9	1.5

Table 1. Summary results from tests of the location and orientation self-calibration service, compared against ground truth measured using 1 cm precision surveying equipment. Each test was performed outdoors, covering approximately 80x50 m area, in two different environments. The CY1 and CY2 tests were deployed in an urban courtyard, while the JR tests were deployed in a forested environment at the James Reserve. Orientation errors are in degrees.

4 Self-calibration Service

While the Acoustic ENSBox self-calibration service is a crucial part of supporting deployable source localization applications, it also serves to exercise and test many of the facilities built into the platform. In this section we describe the design and implementation of the self-calibration service, while showing how it makes use of lower level platform features.

4.1 System Overview

The Acoustic ENSBox self-calibration service enables the implementation of outdoor source localization applications by estimating the 3D locations and orientations of each node in a system of acoustic arrays. Typical source localization algorithms work by computing bearing estimates at multiple points and combining them to estimate a probable location.

However, as we can see from Figure 1, in order to properly interpret and combine bearing estimates, the relative positions and orientations of the sensors must first be known. Furthermore, error in these position and orientation estimates effectively increases the error in the bearing estimates, and thus directly affects the quality of the source localization algorithm. From these requirements of our target applications, we can define the calibration problem, and derive some system performance targets.

4.1.1 *Definition of the Self-calibration Problem*

The acoustic array self-calibration problem seeks to determine a set of parameters that define the locations and orientations of a collection of arrays. The parameters are referenced to the coordinate system specified in the array geometry diagram shown in Figure 2. These parameters are defined in a global coordinate system that can be referenced either to a single origin array, or to a coordinate system defined by one or more arrays placed at surveyed locations.

- Let (X_i, Y_i, Z_i) be the location of array i, relative to the global coordinate system.
- Let Θ_i be the "yaw" orientation of array i, relative to the positive X-axis of the global coordinate system.
- We assume that all arrays are leveled, so that the remaining two degrees of freedom are 0 relative to the global coordinate system.
- If two or more arrays are placed at known coordinates, we add a global scaling variable V that allows the system to scale to fit that coordinate system. This scale factor accounts for various environmental parameters that affect the speed of sound.

We propose a solution that first estimates range and bearing information through acoustic ranging and then uses that information to estimate these parameters. In the next section, we derive a performance goal to support our intended source localization applications.

4.1.2 *System Performance Targets*

Acoustic source localization applications (Figure 1) rely on translating local bearing estimates into a global coordinate system. Errors in the estimates of array position and orientation introduce error in that translation, likely increasing the error in the source localization result. Error in a node's orientation estimate directly offsets the bearing to the source. Error in a node's position estimate can also offset the bearing to a source: for example, a 10 cm error in position amounts to a 0.2 degree error in bearing, for a source perpendicular to that error, at 30 m range.

By considering these errors in the context of the accuracy of source localization bearing estimates, we can derive a performance target. Using similar acoustic arrays, source bearing estimates have been reported accurate to ± 2.5 deg [37]. To reduce the significance of our localization error relative to that figure, we set a performance target of ± 1 deg orientation error and 10 cm 2D position error, representing at worst about 50% of the error in source bearing estimates.

4.1.3 Introduction to the Self-calibration System

The Acoustic ENSBox self-calibration system is initiated by a user during deployment, and is controlled through an embedded web interface. The user deploys the system so that the nodes have consistent radio connectivity and can form a connected multihop network, and such that the nodes will determine a well-constrained set of range relationships. Using the web interface, the user can enter the locations of any nodes that have known locations (measured through out-of-band methods). Then, the user initiates the localization process and observes the results. If the system does not converge well, the user can "refine" the result by triggering additional ranging, or by adding new nodes to better constrain the system. As a rule, in order to achieve good results, each node should have ranges to other nodes that constrain it in orthogonal directions, and three ranging trials are sufficient to acquire all available ranges.

Behind this interface are two subcomponents: an acoustic range and bearing estimation module, and a position estimation module. The range and bearing estimation module measures the range and bearing to other nodes in the system based on the reception of coded acoustic signals called "chirps". The position estimation module triggers each node in the system to "chirp" in turn, and collects the resulting measurements. It then implements a multilateration algorithm to estimate the calibration parameters $(X_i, Y_i, Z_i, \Theta_i)$ as a purely relative coordinate system. Finally, if the locations of some of the nodes are known, the relative system is fit to match those known locations, allowing rotation and uniform scaling.

4.2 Range and Bearing Estimation

The acoustic ranging component of the self-calibration system is an active, cooperative ranging system [32]. Ranging throughout the system is composed of a series of trials in which one node emits a 1/3 second audible ranging "chirp" and other nodes in the system detect that "chirp".

During a trial, the emitter plays the chirp from its speakers and concurrently listens to detect the exact time at which the chirp started. The detected chirp time is enclosed in a chirp notification packet and sent through the flooding service to arrive at the other nodes. Because the flooding service performs time conversion as the message passes through the network, receiving nodes will receive a packet containing a local timestamp. Using the data sampling service, they extract a historical segment of audio data beginning at the

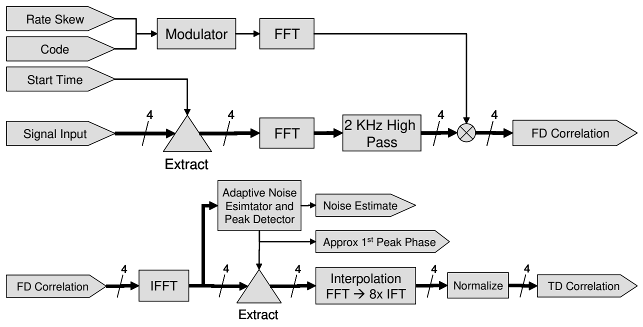

Figure 3. Block diagram of the filtering, correlation, and detection algorithm.

start of the chirp, and queue it for processing. The processing detects the signal, estimates bearing, enhances the signal using beamforming, and performs a final peak detection to compute a time-of-flight estimate.

The sampling, synchronization, and network facilities of the Acoustic ENSBox yields a substantial simplification of the implementation of the ranging system. The ability to retrieve sensor data post-facto with precise timing enables the sender to send a single notification message, after the exact phase of the emitted chirp has been detected locally.[2] Similarly, the flooding service and the time conversion service enable the acoustic ranging code to focus on signal processing problems, with assurance that the samples extracted at the receiver are tightly synchronized with the time of the chirp emission.

In the next few sections, we discuss the processing stages in more detail.

4.2.1 Wideband Audible Acoustic Ranging

The ranging system uses wideband, audible "chirp" signals that have excellent interference rejection properties and phase detection accuracy [13]. The chirp signals are generated from a family of chaotic pseudonoise (PN) codes generated by repeated evaluation of the logistic equation,

$$x_{n+1} = Rx_n(1 - x_n), \quad (1)$$

where $R = 3.98$, $0 < x_0 < 1$ is the seed for the code, and bit n of the code is 1 if $x_n > 0.5$, and 0 otherwise.

These types of chaotic codes have been used successfully in other communications systems, including underwater acoustic communications systems such as [3]. By testing different seed values, we selected the best 128 codes from this family, selecting for low off-peak autocorrelation and low cross-correlation with other codes in the family. In tests we found that the codes were resilient to collisions: detection succeeded even in cases where three different senders chirped concurrently.

The ranging signal is composed of a 2048-chip code, modulated using binary phase shift keying (BPSK) on a 12 KHz carrier. This modulation results in a signal that is spread with the primary lobe covering 6–18 KHz (see [10] for more detailed information, including spectrum plots).

[2]Local detection is required because the sound hardware does not have a facility for causing the sound to start precisely at a predetermined time.

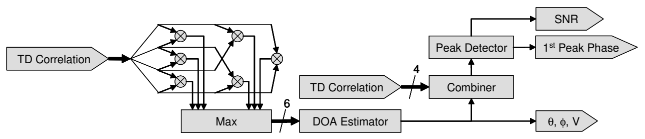

Figure 4. Block diagram of the bearing estimation algorithm.

4.2.2 *Detecting the Ranging Signal*

Figure 3 shows a block diagram of the filtering and detection process. The ranging signal is detected using a "matched filter" implemented by correlation with a reference signal. The reference signal is a copy of the signal originally emitted that is constructed locally based on the code index provided in the notification message.

To detect, the input data is passed through an FFT and pre-filtered to remove low frequency components caused by wind. Next, it is correlated with the reference signal, and the correlation function is then returned to the time domain and analyzed to determine the earliest "spike" in any of the four channels.

A "spike" is detected by first computing running mean and variance estimates using an exponentially weighted moving average (EWMA). The EWMA is initialized based on the 100 samples *prior* to the chirp emission time, which presumably represent ambient noise. Based on the running mean and variance, the algorithm selects as a spike the first point at least 6 standard deviations above the mean. If no point qualifies as a spike, the detection is considered to have failed on that channel. If at least one channel detected the signal, the earliest detection is used and the process continues on to bearing estimation stage.

4.2.3 *Bearing Estimation*

Figure 4 shows the bearing estimation stage, based on Time Difference of Arrivals (TDOA). This algorithm works by exploiting the fact that the signal will arrive at the different points in the array at different times, depending on the direction of arrival of the ranging signal. Thus, by cross-correlating pairs of channel inputs, we can estimate phase lags and fit those lags to the known geometry of the array.

It is important to note that at the sample rate of 48 KHz, each sample corresponds to 0.71 cm of distance. This means that the quantization error due to sampling is a significant fraction of the size of the array (8 cm square). This would result in bearing errors of up to 5 degrees for angles nearly perpendicular to the chord between a pair of microphones.

To address this, the final stages of Figure 3 interpolate the region of the correlation function that surrounds the earliest detection. We interpolate by computing the Fourier coefficients for that region and evaluating them at 8x temporal resolution. This feeds into the start of Figure 4, where we cross-correlate the interpolated segments in the time domain to find the lag at which the correlation between each pair of channels is maximized. Because of the interpolation, these lags can be fractions of a sample. This technique is equivalent to prior work that performed an exhaustive angular search using fractional phase shifts in the frequency domain [4], but our method is much more computationally efficient for our application.

Once the lags are computed, we use least squares minimization to fit them to the array geometry, solving for the azimuth and zenith angles θ and ϕ (see Figure 2). The constraint equations in this system are nonlinear functions that compare the lags between each pair of microphones with a projection of the bearing vector onto the array geometry. We use gradient descent to solve the system and produce the most likely estimate of θ and ϕ.

The technique of first finding lags and then fitting them to the geometry has the advantage that it is resilient to minor deviations in the geometry of the array caused by slightly misplaced microphones [10]. One disadvantage of this method is that only the location of the maximum correlate is used in the fit. This means that this technique won't work well when there are multiple sources of similar energy level, because some pairs may choose one source as the max while other pairs choose another source. However, for our system this is almost never a problem since after the matched filter our signals are located within a tight temporal bound and other interfering sources are strongly attenuated.

4.2.4 *Signal Enhancement via Beamforming*

Once a bearing estimate is computed, we can use this estimate to enhance the ranging signal using a technique called *beamforming*. This technique phase-shifts the input channels to be consistent with the bearing estimate and sums them to compute a single, enhanced channel.

In our implementation, we take into account the *actual* observed lags in addition to the theoretical phase offset derived from the bearing estimate and the nominal array geometry. If the actual lags differ slightly from those computed based on the nominal geometry, the actual lag is used in place of the computed lag. This improves the signal enhancement by accommodating slight deviations in actual microphone placement relative to the nominal geometry.

The enhanced signal is then used to compute the final range estimate by a similar estimator as was used in §4.2.2.

4.3 Position Estimation

The position estimation module drives the self-calibration process by triggering ranging tests, collecting the range and bearing estimates, and implementing a multilateration algorithm to resolve those range and bearing estimates into position and orientation estimates. In this implementation, the multilateration algorithm is centralized, although the "master" node is chosen dynamically and the computation of the ranging and bearing estimation algorithms is distributed. Prior work [6] [34] [20] [21] on distributed multilateration algorithms might be applied, although we have not considered that in this paper.

4.3.1 *Driving the System and Triggering Ranging*

Since our system is intended in part as a deployment tool, our implementation is user-driven. Although this solution is less "automated", in practice this enables the user to help the system by observing and correcting problems that would otherwise be difficult or impossible to correct autonomously.

Once activated by the user, the system triggers ranging by sending a special flooded trigger message that schedules the nodes to emit ranging signals sequentially. The resulting range and bearing estimates are published by each node individually via the reliable publish-subscribe primitive, and the "master" node subscribes to receive these updates. After 30 seconds with no updates, the "master" begins the position estimation algorithm on the new data, and presents it to the user when that algorithm completes.

4.3.2 Non-linear Least Squares

To compute the position estimates we use a non-linear least squares solution based on gradient descent. Our solution is similar to other least-squares solutions such as [6], and to multi-dimensional scaling approaches such as [35] [6] [20] [17] [30]. Our solution differs from pure multi-dimensional scaling in that we use the bearing estimates as well as range to estimate position. Bearing estimates are also used in [6] in their "$r-\theta$" formulation. However, we found that solution performed poorly because the impact of bearing error scales with inter-node spacing, whereas the impact of error from range-based constraints is constant [10]. In addition, our position estimation algorithm has the novel feature of estimating array orientation as well as position.

In our solution, we first check the input data to remove inconsistencies, then iterate: construct a system of constraints and an initial "guess", solve the system, and if any outliers are present, remove them and re-solve. We now detail each of these steps:

4.3.2.1 Consistency Check

In the first step, forward and reverse paths are checked for consistency, and inconsistent data is discarded. While it is often the case that the forward and reverse ranges might differ by a few cm, large differences are a sign of a detection error. For example, these differences can arise in instances where the line of sight (LOS) path is partially obstructed and attenuated relative to a reflected path. Although the forward and reverse path attenuation is symmetrical, a source of interference near one receiver can cause an asymmetric measurement if the attenuated LOS signal is below the noise threshold of one, but not both receivers.

To address these cases, we use the heuristic to accept the shorter range and drop all information about the longer range; since if the long range is caused by a reflection, the bearing estimate is likely to also be incorrect.

4.3.2.2 Initial Guess

The gradient descent algorithm requires an approximate starting point, from which it will refine. To compute this initial guess, we consider one node as the origin and use the range and bearing estimates to derive positions and relative orientations for its neighbors. Initial orientation estimates are determined by "looking back" from a newly positioned node towards its source. The forward and reverse bearing estimates should differ by 180 deg, so the difference from that is accounted as a difference in relative orientation. As the coordinate system grows, node position and orientation estimates can be computed from multiple nodes and averaged.

4.3.2.3 Constructing a System of Constraints

Between each pair of nodes, we can formulate three independent constraints: range, azimuth and zenith constraints. In these constraints, we consider the orientation Θ_i of each node to be a constant value that is estimated separately.

The range constraints are a formulation of multi-dimensional scaling, stipulating that the distance between two nodes i and j must equal the smaller of the two measured ranges $R_{i,j}$:

$$R_{i,j} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}. \tag{2}$$

The azimuth constraints use the arctangent function to relate the azimuth estimate $\theta_{i,j}$ to the node coordinates:

$$\arctan \frac{Y_j - Y_i}{X_j - X_i} + \Theta_i = \theta_{i,j}. \tag{3}$$

The zenith constraints relate the Z dimension to the observed zenith angles $\phi_{i,j}$:

$$\arctan \frac{Z_j - Z_i}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}} = \phi_{i,j}. \tag{4}$$

4.3.2.4 Solving the System

In order to solve these constraints using gradient descent, they must be "linearized", or differentiated in terms of the variables (X_i, Y_i, Z_i) to form a Jacobian matrix.[3] Once linearized, the system can be solved using iterative gradient descent: the variables are initialized with our initial guess, and the linearized system is solved to compute a correction to the positions, iterating until the corrections drop below a defined tolerance.

4.3.2.5 Estimating Orientation

In our formulation of the azimuth constraints, we did not include the node orientation Θ_i as a variable, because including it prevented the system from converging. Since the node orientations Θ_i are not variables in the azimuth constraints, we must refine our orientation estimates separately. To do this, we recompute the orientation estimates after each update of the position estimates, by computing the average residual value for the azimuth constraints for each node i, using a vector sum. If we assume that there are no outliers, this average residual angle represents a correction to Θ_i that will zero the average residual computed this way.

Solving the orientation and positions separately has the disadvantage that a change in one set of variables can counteract a change in the other, resulting in a failure to converge. To address this, we stop updating the orientation values after 10 iterations, and allow the positions to adjust until convergence.

4.3.2.6 Outlier Rejection Heuristics

Our self-calibration system tends to result in systems that are sufficiently over-constrained to support detection and rejection of inconsistent data. We apply several heuristics to identify and reject inconsistent data that would otherwise just add to the estimation error. Our heuristics are based on the observation that one of the most likely sources of significant error is the detection of a reflected path when the line-of-sight (LOS) path is severely attenuated.

These errors tend to have two properties. First, whereas typical range errors in LOS conditions are under 10 cm,

[3]Due to space limitations we omit these equations, but they can be found in [10].

in non-LOS conditions reflections can introduce arbitrarily large range errors, often meters or tens of meters. Second, reflections generally produce errors in bearing estimates, since the reflected path often arrives from a different direction than would a LOS path. We use two methods to identify and reject inconsistent data.

First, during the node orientation estimation described in §4.3.2.5, a severe angular inconsistency will appear as an outlier in the distribution of azimuth residuals. After allowing the system to partially converge, we drop constraints that are marked by an inconsistent angle, and continue to iterate to convergence.

Second, after the system has fully converged, we use the method of *studentized residuals* to weight the residual errors by a measure of how much they impact the system. This technique has been used in a similar way in the Active Bat system [38]. If any weighted residual exceeds a fixed threshold, the constraint with the largest weighted residual is dropped and the system is recomputed.

4.3.2.7 Costs and Convergence Properties

The position estimation algorithm runs centrally on one of the nodes. Although the cost of running the algorithm can be high, this is not of inordinate concern because this algorithm typically runs only once per deployment.

The Jacobian matrix that must be solved is a $3(N-1)$ by $3R$ matrix, where N is the number of nodes and R is the number of pairs of nodes with valid ranging data. The iterative least squares algorithm runs for a minimum of 10 iterations to allow the orientation estimates to settle, and will then continue until the convergence condition is met. Typically only a few more iterations are required; if 100 iterations are run convergence is considered to have failed. Outlier rejection causes additional costs because the entire algorithm is re-run after each constraint is dropped.

In the courtyard experiments described in §5.3.1, N was 10 and R was approximately 70. In these tests, each pass through the solver took an average of 96 seconds on the ARM processor. On average one outlier was dropped, yielding an average run-time of 216 seconds.

As the number of nodes scales up the cost of solving the Jacobian matrix will grow as $O(N^3)$. Since we currently have only 10 nodes, we have not attempted to address this issue. However, in prior experience with similar algorithms [24], we have been able to scale to larger networks by solving the system incrementally, adding a few nodes at a time.

At the present time, we have not analyzed the convergence properties of this system. In general, convergence depends on the degree of range connectivity, the geometry of the network, and the presence of outliers. In practice we have found that the system has always converged when more than 4 nodes are present and each node has at least four ranging neighbors.

4.3.3 Association to Survey Points

The output of the position estimation algorithm will be a self-consistent *relative* position and orientation map, with a scale relative to the speed of sound. The speed of sound in air is not a constant, but rather a function of environmental parameters, primarily temperature and humidity. While relative coordinates may be sufficient for some applications, many applications will need to relate this map to real-world coordinates such as GPS.

Most localization schemes address this by specifying certain nodes to be "anchors" with exact known locations, and building the coordinate system around them. While this approach has natural application in distributed localization schemes, it introduces warping if the range data is not properly compensated to correct the speed of sound. Temperature compensation is subject to measurement error, both because air temperature sensors typically have accuracy limited to about 0.5 degree C, and because of the difficulty of properly shielding the sensor to get a clean reading. Since scaling from temperature can be as much as 0.2% per degree C, at 80 m range the error from a 0.5 degree offset would be 7 cm, doubling our typical detection error of 3 cm.

Instead, we solve the coordinate system matching and scaling problems at once by *fitting* the purely relative position estimates to a few known node positions. To implement this post-facto fit we apply techniques modeled on Procrustes shape matching [7]. We apply a four step process to fit the estimated map to the surveyed map:

- **Filter Scale.** Over all pairs of survey locations, we separately sum the estimated and the actual distances, and derive a scale factor by computing the ratio of the two sums. We then scale the estimated map by that factor.
- **Filter Translation.** We translate the maps so that the node closest to the centroid is the origin in both maps.
- **Filter Rotation.** We compute the 3D rotation about the central point that results in the closest match.
- **Final Translation.** Finally, we apply these transforms to the entire estimated map, and then apply a final translation to match the survey coordinate system.

Not only does this approach enable applications to properly interpret the output of the position estimator, it also serves as a good way to compare our results to ground truth in our performance metric.

5 Experiments and Results

To evaluate the performance of our self-calibration system we performed a number of experiments. We performed two types of experiments: *component tests*, in which we tested ranging and bearing performance using a controlled test environment, and *system tests*, in which we performed end-to-end tests of the whole system, measuring the accuracy of our position and orientation estimation in several different target environments. In all of these tests we placed great importance on presenting the system with a realistic environment. For our platform to succeed it must successfully self-calibrate in a variety of environments determined by the applications.

The environment can present a number of challenges to an acoustic ranging system, including noise, obstructions and reflections from clutter, and weather and environmental conditions. Indoor environments tend to be fairly quiet, but pose challenges from reverberations and reflections. The outdoor acoustic environment is often quite noisy, suffering from wind noise and different types of background noise in different environments. Interesting outdoor environments

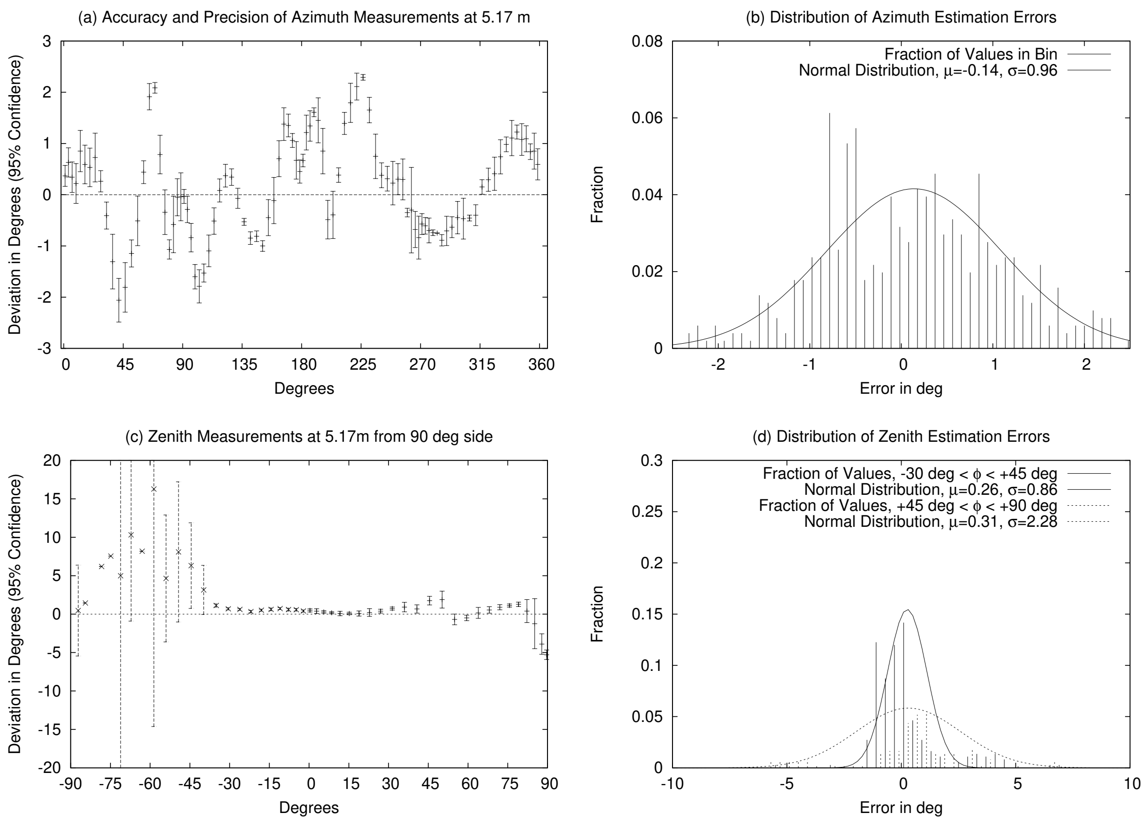

Figure 5. (a) and (b) show the accuracy of the bearing estimator as a function of azimuth and zenith. (c) and (d) show the overall distribution of errors in azimuth and zenith bearing estimates. In (d) we show separate distributions for the "midrange" and "overhead" zenith angles.

typically contain clutter near the ground, which can block or attenuate signals and introduce reflections. Environmental factors such as wind, temperature, and humidity have an impact on acoustic time-of-flight systems, as we discussed in §4.3.3. For our tests, we performed controlled component tests in enclosed environments, while performing system tests in outdoor environments, both urban and forested.

In the next sections we describe each of our experiments in detail, beginning with bearing and range component testing, and then discussing each of the system tests.

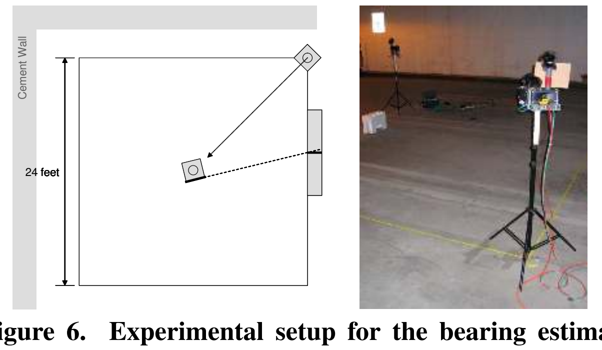

Figure 6. Experimental setup for the bearing estimation component test. To measure angular ground truth, a laser aligned with the receiver in the center is pointed at a measured location along the edge of the square.

5.1 Bearing Estimation Component Testing

To test bearing estimation, we set up the experiment shown in Figure 6 to record bearing estimates with carefully measured ground truth. The test was performed in an parking structure with significant reverberation and 65–70 dB of background noise. The emitters were calibrated to chirp at 100dB sound pressure level (SPL) at 1 meter. We performed two tests: an azimuth test in which we rotated the receiver through 360 degrees, and a zenith test in which we turned the receiver on its side and rotated it 360 degrees to emulate signals coming from a variety of elevations.

Figure 5 shows the results of the bearing component test. Figures 5(a) and 5(b) show the precision and accuracy of the bearing estimator as a function of azimuth, with zenith 0. While azimuth performance is overall quite good, the results show a dependence on bearing angle. We do not have a definite explanation for this, although we hypothesize that it is caused by slight deviations in the placement of the microphones in the array, and by cases in which the array itself obstructs the signal.

Figures 5(c) and 5(d) show the precision and accuracy of the bearing estimator as a function of zenith angle, with azimuth 90. Note that for zenith angles less than -30 degrees, the signal is highly obstructed, because it must pass through or around the base of the array. These obstructed cases are highly inaccurate, but they are also fairly rare in typical deployment scenarios. For zenith angles above -30 degrees,

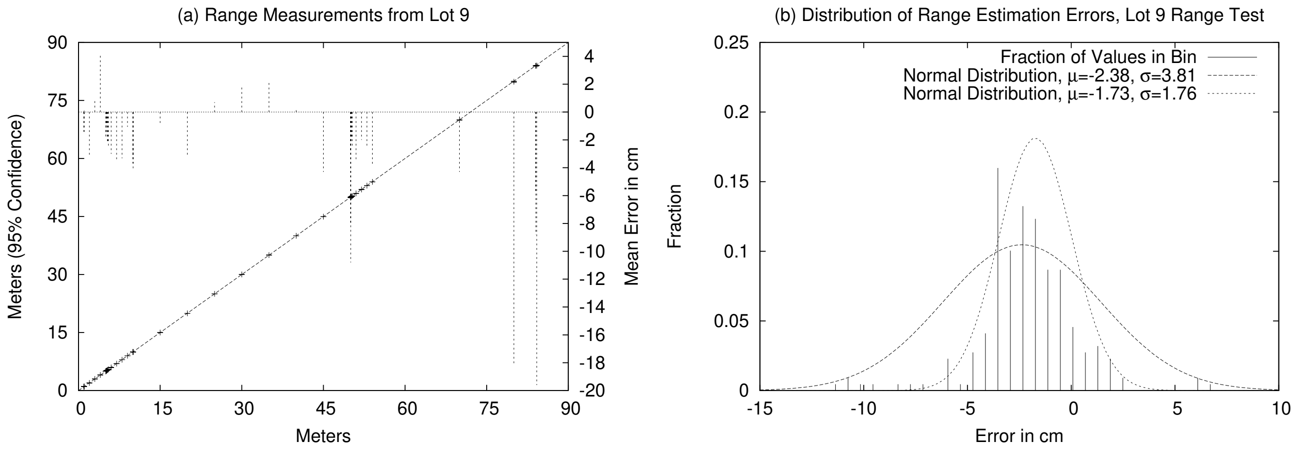

Figure 7. (a) Shows range estimation error relative to ground truth, for our sequence of range experiments. The bottom axis shows the ground truth distance for an experiment, while the left axis shows the average estimate, with 95% confidence intervals. The right axis details for each experiment, the deviation of the mean estimate from ground truth. (b) Shows the distribution of errors. The dotted curve shows the fit to a normal distribution if the 17 values with error larger than 10 cm are dropped.

the results are quite accurate, especially in the "midrange" region from -30 to +45 degrees, where the typical results are within 1 degree of the correct bearing.

5.2 Range Estimation Component Testing

The next experiment performs a controlled test of the performance of the ranging component. This test was performed in the same parking structure used for the bearing test, and approximately the same environmental conditions, and the emitters were calibrated to chirp at 100dB SPL at 1 meter. For this test, the receiver was fixed and the emitter was carefully moved along the ground, using a laser range-finder and measuring tape to establish ground truth.

Figure 7 shows the results of this experiment. To get a clearer picture of both accuracy and precision, we performed range tests at a variety of distances and scales, performing clusters of tests at 1 m, 5 m, 10 m, and 50 m. In Figure 7(a) we highlight the deviation from ground truth as dashed impulses. We observe that with the exception of a few measurements at 50 m, the magnitude of the error is consistently less than 5 cm.[4]

Beyond 75 m, the error increased substantially, largely because the ranging signal began passing out of the detection window. The system uses a recording window of 16K samples and a ranging signal of 8K samples, meaning that after 58 m the ranging signal will begin to extend beyond the recording window. In other tests, we have extended the system range to about 120 m by doubling the recording window, while incurring a higher computation cost for detection.

5.3 System Tests

To evaluate the performance of the entire system, we performed tests in two environments: outdoor urban and outdoor forested. In these tests, we measured ground truth position as accurately as we could, using professional surveying equipment with a 3D accuracy of under 1 cm. For the outdoor urban test we also used a laser to align each array to point at some other array, in order to get both accurate bearing measurements and a diverse set of bearings. For other tests we used a compass to align the arrays to point approximately north. In each outdoor test, we deployed 10 nodes in an area approximately 50x80 m, and calibrated the emitters to chirp at 100dB SPL.

In order to characterize the performance of our system we must define a performance metric and a method for assessing it. Since we are interested in position error relative to ground truth, we use the algorithm described in §4.3.3 to fit the output of our estimation algorithm to our surveyed position data. We then base our performance metrics on the distances between points in our map and the corresponding ground truth points.

In Table 1, we report several metrics for each test: average and median 2D position error, average and median 3D position error, and average and median orientation error. We report 2D and 3D separately because our deployments tend to be flat, yielding poor constraints in the Z axis and thus much higher error. In addition, our target source localization applications can localize a target in 2D independently of Z position estimates. We also report the distributions of estimation errors to give a better perspective on the repeatability of the estimation system.

5.3.1 Outdoor Urban Test

Two outdoor urban deployments were performed in an outdoor courtyard, shown in Figure 8(a-c). The perspective view in Figure 8(a) shows that the courtyard is a flat, mostly open space containing planters and tall hedges which sometimes obstruct LOS. It is surrounded by brick buildings that are good reflectors. The environment has significant levels of background noise from ventilation fans and nearby roads and construction projects, with sound pressure levels typically in the range of 65–70 dB.

[4]These errors at 50 m resulted from a loss of sync due to a temporary connectivity failure.

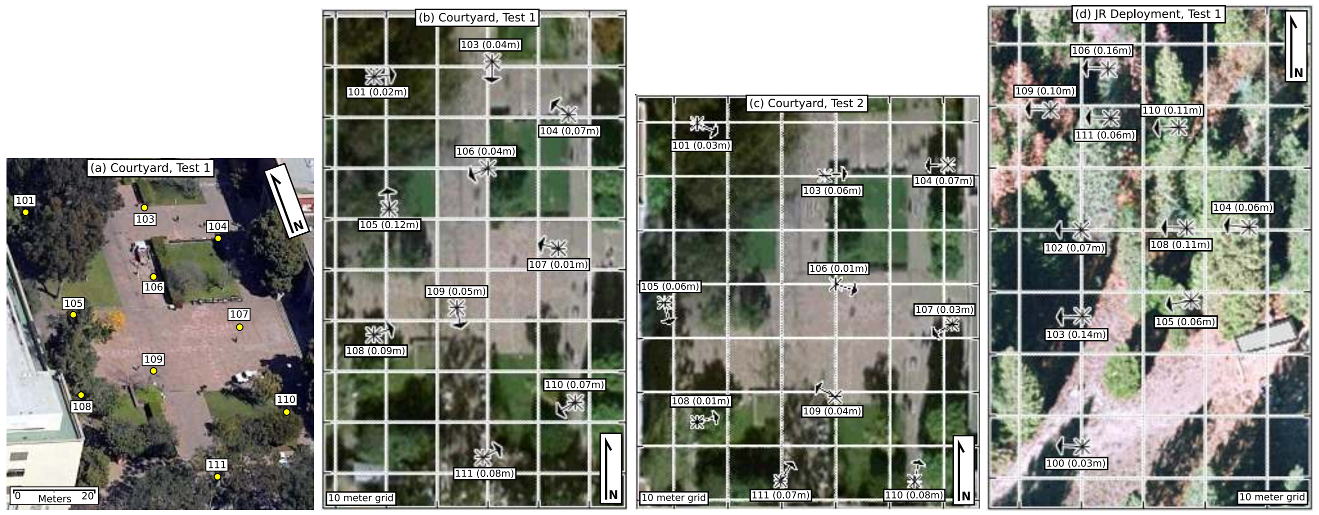

Figure 8. (a) shows a perspective view of the urban courtyard, with reflecting walls and tall, obstructing hedges and planters. (b), (c) and (d) show the experimental setup for our three test deployments: Courtyard 1, Courtyard 2, and the James Reserve in Idyllwild. Ground truth node locations are indicated by the X's. The '+' and arrow indicates a position and orientation estimate from our system, and the 2D position error in m is shown in parentheses. Photo credits: (a) Virtual Earth; (b) and (c) Google Earth; (d) the James Reserve.

We ran four experiments in this environment: two tests in each of the two deployment configurations shown in Figure 8(b) and (c). A map of node positions is overlaid on each photo, where 'X' represents ground truth and '+' represents the estimated position, with the actual error reported in meters. Table 1 summarizes these results as CY1 and CY2.

To show the error in more detail, Figure 9(a) shows an expanded view of the deviation in 2D position for all nodes, from all four courtyard experiments. This graph also shows the error reduction resulting from our outlier rejection heuristics; in some cases the error is reduced by 50%. Note also that because of the flat topology, the performance of the system along the Z axis is much lower than for X and Y.

Figure 9(c) shows a histogram of orientation errors relative to ground truth, with and without outlier rejection. With outlier rejection, the distribution is considerably tighter.

5.3.2 *Outdoor Forested Test*

Figure 8(d) shows our outdoor forested test at the James Reserve in Idyllwild, CA. We planted 10 stakes in a 50x70 m region, using a compass to align the arrays facing west.

The forest environment is much more complex than the urban environment. The terrain is hilly, varying 5 meters from lowest to highest point, with significant foliage and clutter near the ground. Grass, bushes, and trees obstructed LOS between many pairs of nodes.

However, as we can see from Table 1 and Figure 9, this more complex environment causes only limited degradation of the localization performance. From courtyard to forest, the 2D performance degraded from an average error of 5 cm to 9 cm. Likewise, the accuracy of the orientation estimates degraded from an average of 1.3 degrees error to 2.7 degrees.[5] However, because of greater height diversity in the forest terrain, the system was better constrained and the 3D performance improved from 45 cm to 33 cm.

[5]The apparent degradation of orientation accuracy at JR may partly be due to errors in ground truth. The arrays were aligned "by eye" with a small hand-held compass.

5.3.3 *Effect of Obstructions and Reflections*

While our tests did include cases of partial obstruction, the line of sight (LOS) path was rarely *completely* obstructed. Figure 10 shows results measuring the impact of adding simulated obstructions and reflections to the CY1 urban courtyard data set shown in Figure 8(b). The graphs are cumulative distribution functions (CDF)s over all possible node pairs, showing the maximum position error observed when a single selected node pair is subject to a simulated reflection, with varying parameters. The data show that: (1) our rejection heuristics are more likely to work in better-constrained systems, (2) obstructions that block LOS yield less constrained systems and higher error, and (3) reflections accomanied by large angular deviations are rejected more readily, even with fewer constraints. We now detail the setup for each of the three simulations.

CY1 has data for 32 node pairs. For each pair in turn, we compute the mean position error induced if *only* that pair registers a reflection. The middle curve shows a CDF of mean position error, in which one selected path is extended by 10 m; the bad range is rejected in 75% of the cases.

For the other two curves, we additionally simulate a 10x10 m obstruction in the center of the field, completely blocking 11 of the 32 pairs. The lower curve shows the CDF when a single range (possibly one of the 11 blocked ranges) is extended by 10 m. With the simulated obstruction the system is less constrained; it can correctly reject the bad range only 50% of the time. The top curve shows the CDF for the obstructed case when the bad range also includes a 45 degree angular error. Because in these cases the angular inconsistency heuristic applies, the bad range is correctly rejected in all but one case.

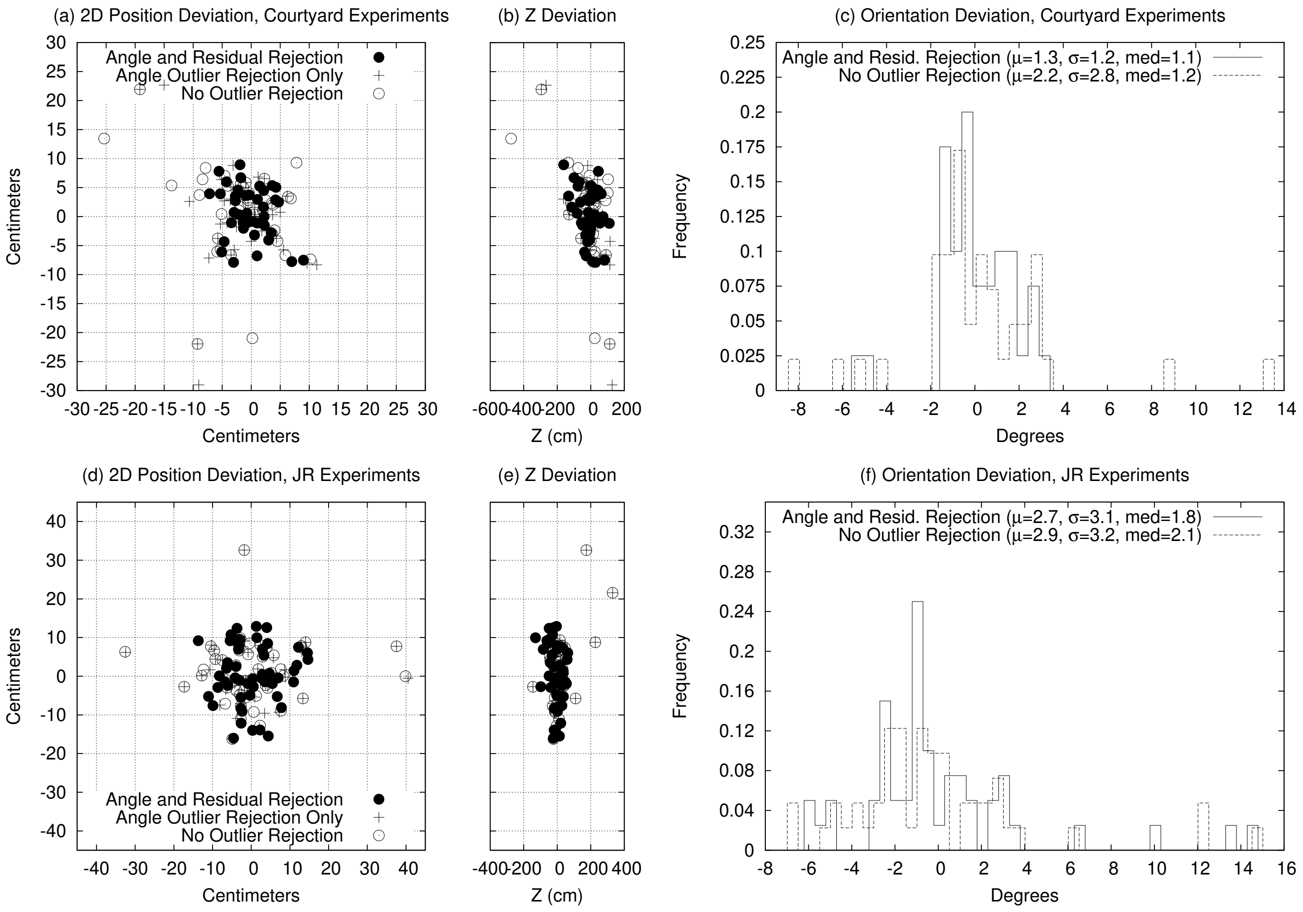

Figure 9. Shows the distribution of deviations from ground truth for our two test environments. The upper set of graphs shows our system performance in the urban courtyard tests, while the lower set shows our performance in the James Reserve. (a) and (d) show the deviation of our 2D position estimates from ground truth, for all 10 nodes in the four courtyard experiments. If all positions were estimated perfectly, all points would be at (0,0). The three types of point show the improvement resulting from our outlier rejection heuristics. (b) and (e) show the deviation of Z estimates, vs. the same Y axis. (c) and (f) show histograms of the deviation of our orientation estimates relative to ground truth, with and without angular outlier rejection. In graph (d), one additional outlier (65,21) was left out of the plot.

In general, resilience to reflections is a function of the degree of constraint, which is in turn a function of geometry and range connectivity. While these simulations show that our rejection heuristics are promising, more experiments are needed to gain a better understanding of the true impact of reflections and obstructions and to characterize the typical error in angle of arrival due to reflection.

6 Discussion

In the preceding section we presented some detailed results from our component and system testing in several outdoor deployments. We now compare those results to some of the related work that we described previously.

Our ranging precision after temperature compensation is generally 5x better than the other audible acoustic systems from Sallai [31], Kwon [20], and Kushwaha [18]. Sallai and Kwon do not cite variance estimates, but their graphs show standard deviations of at least 15–25 cm, depending on whether or not outliers are rejected, compared with 1.7–3.8 cm for our system. However, this is not a fair comparison, as these systems are based on much more resource-constrained Mica2 hardware.

A more fair comparison can be made to Kushwaha [18], who presents a system based on the Vanderbilt counter-sniper platform [22]. The Kushwaha system uses a similar approach to ours, substituting a linear frequency sweep in place of our PN code. It employs a matched filter and repeated, position-modulated chirps to enhance SNR in the detector. For ranges in excess of 10 m, Kushwaha cites standard deviations after outlier rejection of about 25 cm, compared with 1.7 cm for our system. This difference may be due to the higher process gain and lower autocorrelation noise exhibited by our PN codes; on the other hand, our matched filter is likely more expensive.

In addition to improved precision, our ranging system has a longer detection range for equivalent power output. In a test calibrated to match those of Kwon and Kushwaha (105 dB SPL at 10 cm),[6] our system achieved 60 m range,

[6]Our other tests achieve higher range, but output 100 dB at 1 m.

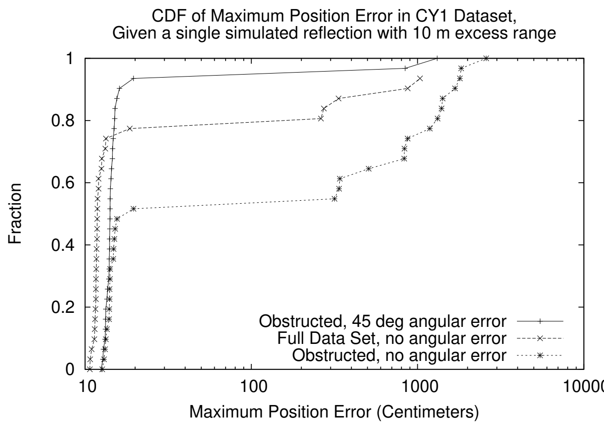

Figure 10. CDFs taken over all possible pairs in the CY1 data set, showing the maximum position error observed when a single selected range pair is subject to a simulated reflection, with varying simulation parameters.

2x better than the 20–30 m cited in Kwon and Kushwaha. We believe this difference in range is primarily a function of the coding in our ranging signal and the process gain in our detector. In addition, longer detection range results in a better constrained system and thus lower positioning error at any given deployment density.

The results of outdoor system tests also showed improvement over prior work. In an outdoor urban test, our average 2D position error is 50x lower than Kwon's reported 2.46 m error (10x lower when additional simulated ranges are added to result in 0.48 m). In the more challenging forested environment our average error doubles, although it is unclear how the other systems might perform under similar conditions. This improvement is achieved in spite of estimating the additional variables of vertical height and orientation. Our results also compare favorably to the simulation results from Kushwaha, who reported a mean 3D position error of 89 cm; compared with 33 cm from our forest experiment.

This work represents a significant improvement in the accuracy of self-localization relative to other work, at a corresponding higher computational cost. However, the key advantage of the Acoustic ENSBox is that it also serves as a general-purpose acoustic sensing platform: the enhanced computational resources that enable more accurate self-localization also enable easy prototyping of a wide variety of localization and classification applications.

7 References

[1] A. Arora et al. A line in the sand: A wireless sensor network for target detection, classification and tracking. *Computer Networks*, 46(5):605–634, Dec. 2004.

[2] A. Arora et al. Exscal: Elements of an extreme scale wireless sensor network. In *IEEE RTCSA*. IEEE, 2005.

[3] S. Azou, C. Pistre, and G. Burel. A chaotic direct sequence spread-spectrum system for underwater communication. In *Proceedings of the IEEE Oceans Conf.*, Biloxi, Mississippi, Oct. 2002.

[4] P. Bergamo et al. Collaborative sensor networking towards real-time acoustical beamforming in free-space and limited reverberence. *IEEE Transactions on Mobile Computing*, 3(3):211–224, July–September 2004.

[5] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05)*, pages 31–42, August 2005.

[6] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal. Ad-hoc localization using ranging and sectoring. In *IEEE INFOCOM*, 2004.

[7] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.

[8] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *OSDI*, pages 147–163, Boston, MA, December 2002.

[9] J. Elson and K. Romer. Wireless sensor networks: A new regime for time synchronization. In *First Workshop on Hot Topics in Networks (HotNets-I)*, 2002.

[10] L. Girod. *A Self-Calibrating System of Distributed Acoustic Arrays*. PhD thesis, Univerity of Caliornia at Los Angeles, 2005.

[11] L. Girod, V. Bychkovskiy, J. Elson, and D. Estrin. Locating tiny sensors in time and space: A case study. In *in Proceedings of ICCD 2002 (invited paper)*, Freiburg, Germany, September 2002.

[12] L. Girod., J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *USENIX*, 2004.

[13] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IROS*, Oct. 2001.

[14] L. Girod et al. A reliable multicast mechanism for sensor network applications. in CENS Technical Report 48, 2005.

[15] B. Greenstein et al. Capturing high-frequency phenomena using a bandwidth-limited sensor network. In *ACM SenSys*, Boulder, CO, Nov 2006.

[16] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov/Dec 2002.

[17] X. Ji and H. Zha. Sensor positioning in wireless ad-hoc networks with multidimensional scaling. In *IEEE INFOCOM*, 2004.

[18] M. Kushwaha, K. Molnár, J. Sallai, P. Völgyesi, M. Maróti, and A. Lédeczi. Sensor node localization using mobile acoustic beacons. In *The 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, 2005.

[19] B. Kusy, P. Dutta, P. Levis, M. Mar, A. Ledeczi, and D. Culler. Elapsed time on arrival: A simple and versatile primitive for canonical time synchronization services, in press, 2006.

[20] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha. Resilient localization for sensor networks in outdoor environments. In *IEEE ICDCS*, 2005.

[21] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks, special issue on Wireless Sensor Networks*, 2003.

[22] A. Ledeczi et al. Countersniper system for urban warfare. *ACM Transactions on Sensor Networks*, (2):153–177, Nov. 2005.

[23] M. Maroti, P. Volgyesi, S. Dora, B. Kusy, A. Nadas, A. Ledeczi, G. Balogh, and K. Molnar. Radio interfermetric geolocation. In *ACM SenSys*. ACM, Nov. 2005.

[24] W. Merrill et al. Dynamic networking and smart sensing enable next-generation landmines. *IEEE Pervasive Computing Magazine*, Oct-Dec 2004.

[25] W. Merrill, L. Girod, J. Elson, K. Sohrabi, F. Newberg, and W. Kaiser. Autonomous position location in distributed, embedded, wireless systems. In *the IEEE CAS Workshop on Wireless Comm. and Networking*, Pasadena, CA, 2002.

[26] W. M. Merrill, K. Sohrabi, L. Girod, J. Elson, F. Newberg, and W. Kaiser. Open standard development platforms for distributed sensor networks. In *SPIE Unattended Ground Sensor Technologies and Applications IV*, pages 327–337, 2002.

[27] D. L. Mills. Internet Time Synchronization: The Network Time Protocol. In Z. Yang and T. A. Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

[28] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location support system. In *ACM MobiCom*. ACM, Aug. 2000.

[29] N. B. Priyantha, A. K. L. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applicaitons. In *ACM MobiCom*, 2001.

[30] V. C. Raykar and R. Duraiswami. Automatic position calibration of multiple microphones. In *IEEE ICASSP04*. IEEE, 2004.

[31] J. Sallai, G. Balogh, M. Maroti, A. Ledeczi, and B. Kusy. Acoustic ranging in resource-constrained sensor networks. In *ICWN '04*, June 2004.

[32] A. Savvides, L. Girod, M. Srivastava, and D. Estrin. Localization in sensor networks. In C. S. Raghavendra, K. M. Sivalingam, and T. Znati, editors, *Wireless Sensor Networks*. Kluwer, 2004.

[33] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM MobiCom*, pages 166–179, New York, NY, USA, 2001. ACM.

[34] A. Savvides, H. Park, and M. B. Srivastava. The n-hop multilateration primitive for node localization problems. *MONET Special Issue on Sensor Networks and Applications*, 2003.

[35] W. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[36] H. Wang et al. A wireless time-synchronized COTS sensor platform, part II: Applications to beamforming. In *IEEE CAS Workshop on Wireless Communications and Networking*, 2002.

[37] H. Wang et al. Acoustic sensor networks for woodpecker localization. In *SPIE Conference on Advanced Signal Processing Algorithms, Architectures and Implementations*, Aug. 2005.

[38] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5), Oct. 1997.

[39] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli. Blind beamforming on a randomly distributed sensor array system. *IEEE JSAC*, 16(8):1555–1567, Oct. 1998.