Sensor Network Localization Based on Natural Phenomena

by

Daniel Sang Kim

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Electrical Engineering
and Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2006

Author .....
Department of Electrical Engineering and Computer Science
August 12, 2006

Certified by .....
Joseph A. Paradiso
Associate Professor, MIT Media Lab
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Sensor Network Localization Based on Natural Phenomena

by

Daniel Sang Kim

Submitted to the Department of Electrical Engineering and Computer Science
on August 12, 2006, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Electrical Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Autonomous localization is crucial for many sensor network applications. The goal of this thesis is to develop a distributed localization algorithm for the PLUG indoor sensor network by analyzing sound and light sensory data from naturally occurring background phenomena as well as synthesized emulations of background transients. Our approach has two main phases: passive and active. The system enters an active mode when its sensed region stays relatively silent and stable, hence assumed to be unoccupied; otherwise, it stays in the passive mode. In the passive mode, each node looks for sonic transients and compares the timing of its highest sound peak to that of synchronized sound peaks from other nodes in its neighborhood in order to estimate its distance. Passive ranging achieved $50.96cm$ error and simulated passive localization achieved $103.06cm$ error with a typical node-spacing of $2m$. In addition, the system exploits background transients based on light sensory data to determine room boundaries. In the active mode, each node occasionally generates recorded mimics of natural sonic transients, like pencils dropping or water glasses clinking and manipulates an attached light source. Active acoustic ranging achieved $2.1cm$ error and simulated active localization achieved $7.97cm$ error with a typical node-spacing of $2m$. In addition, passive location estimation in a real deployment is found to converge as more sensory data is available; range resolutions of $2.5m$ and localization errors of $20.3cm$ were obtained after running in passive mode for 20 hours in $7m$ by $5m$ dorm hallway. The main features of author's approach are its distributed properties, the lack of any heavy infrastructure, its unobtrusive exploitation of multi-sensory background phenomena, and in active mode, making the sound signal between nodes unobtrusive by mimicking the natural sounds.

Thesis Supervisor: Joseph A. Paradiso
Title: Associate Professor, MIT Media Lab

Acknowledgments

I would like to thank Josh Lifton, Yasuhiro Ono and Joe Paradiso for their guidance and mentorship over the past two years. I also wish to thank Samsung, the Things That Think Consortium and other sponsors of the MIT Media Lab for their generous support. Portions of this work are supported by Samsung Electronics Fellowship.

Contents

Chapter 1

Introduction

In an article titled *21 Ideas for the 21st Century* published in *Business Week* [80], Nobel Laureate Horst Stormer wrote:

> Unfettered micro sensors will go anywhere and measure anything - traffic flow, water level, number of people walking by, temperature. This is developing into something like a nervous system for the earth, a skin for the earth. The world will evolve this way.

Modern technology has become mature enough to create microprocessors and peripherals in miniature size, small enough to embed in almost any object such as shown in Figure 1-1. This chip contains memory, modem, antenna and microprocessor in $2-4mm^2$ size with 512 kilobytes memory. Data can be moved in and out of the chip at speed of up to 10 megabits per second. According to the researchers at HP[4], this chip could be used to ensure that drugs have not been counterfeited or to add sounds or video to postcards, hence enabling digital data to be attached to related physical objects seamlessly.

In addition, the admission ticket for the 2005 World Exposition[75], which has approximately 22,050,000 visitors, employed the Hitachi μ-chip with size $0.15 \times 0.15 \times 7.5\mu m$. Despite its small size, it has an outstanding performance: no incidence of confirmed forgery and 0.001% incidence of ticket recognition error. Weiser's [89] vision for ubiquitous computing and Butera's [11] vision for paintable computing are

becoming true. The environment is becoming *responsive*, being embedded seamlessly with sensor devices.

1.1 The Future with Ubiquitous Computing and Sensor Network

In the future, the author believes that every building will be equipped with *infrastructure sensors* (IS) and everyone will carry *mobile sensors* (MS), which is more capable than current mobile phones. The MS, storing personal information and life patterns, reacts with the augmented "environment" to make everyone's life more comfortable. The MS communicates with other MS's, IS's, and its user while operating at its minimum possible energy. The future MS is not just a cell phone with camera and mp3 player; it will replace credit cards, IDs, admission tickets, paper clips and many other devices. The MS can communicate with other MS's, for example, to avoid car collisions. A relevant project from the MIT Media Lab, the UbER-Badge [52], supports many applications in this space, including current business cards, helping users to exchange information electronically and to transmit automatically all necessary information to the main server, which emails relevant information to the user's account. Digitized paper clips, called DigiClips[16], turn passive paper documents to active physical documents that keep track of changes both of the physical and the virtual document. In addition, the infinitesimal size of the sensor will enable "futuristic" bio applications [90] [55]. There are currently enormous numbers of other applications of sensor networks. However, the greatest use of sensor network lies not only in exiting applications, but also in future applications that cannot be

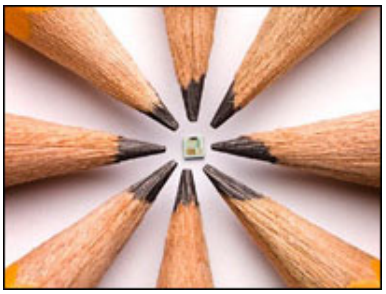


Figure 1-1: HP's Memory Spot chip in its 2-4 square mm area with memory, modem, antenna, and microprocessor. This picture is excerpted from BBC news [4].

imagined yet.

1.2 State of the Art

If someone visits a new city for the first time, the best place to get a grasp of the city's current technology is its market. The market is where most people exchange, sell, and buy their new ideas. Accordingly, the author would like to discuss some current sensor networks in commercial products.

SensorWare Systems [96], a company spun-out of NASA, develops the Sensor Web for environmental monitoring and control. As each node can be orbital or terrestrial, fixed or mobile, sensor web has proven valuable in agricultural, homeland security, and remediation needs. Known for its adoption and promotion of the ZigBee standard, Ember is a another sensor network company founded by MIT Media Lab alumni Robert Poor and Andrew Wheeler. Their applications include home, building, industrial and power automation in addition to asset management and defense. WhereNet [92] uses wireless sensor networks for tracking and managing enterprise assets in automotive, transportation, logistics, aerospace, defense and healthcare.

However, these commercial products are still in a primitive stage, despite many academic studies in sensor networks. Several research projects in hardware design for sensor networks [99][59] [60] have tried to minimize their size while retaining enough CPU power, energy efficiency, and memory. At the Media Lab, Benbasat [6] has designed and constructed a modular platform for use in compact wireless sensing with minimum power. Laibowitz[38] developed small nodes to illustrate parasitic mobility for pervasive sensor networks. With a long history and more hard work, hardware limitations seem to be receding. Although hardware designs are growing more capable, no particular algorithm has had widespread implementation in hardware yet due to many reasons, such as applicability, energy efficiency, sensor data fusion, etc. The sensor network literature has exploded in recent years, littered with many test applications, node design protocols, and software architectures. One of the first notable application papers is [50], which provides an in-depth study of applying wireless

sensor networks with 32 nodes to real-world habitat monitoring, streaming live data onto the web. Other applications [69][61] focus on resource-efficient protocols in medical settings. Based on coarse and primitive sensor data with statistical manipulation, Wren [98] shows how to recover information about sensor geometry. Werner-Allen [91] applied sensor networks to volcano eruption monitoring and evaluated its approach in terms of energy, bandwidth usage, and accuracy of infrasonic signal detection.

1.3 Motivation: PLUG and Localization

Although applications of sensor networks have been studied for many decades, their current commercial applications stay fairly primitive, as pointed out in the previous sections; this means that there is still much work to do. Due to limited resources, sensor network applications pose many challenges [20]. These challenges include time synchronization, power management[6][5], memory management, security issues[63][77], sensor data fusion[49][48], localization[73], middleware and so forth. Among the notable middleware projects are CodeBlue[47], Mate[42][43], Agilla[23], Region Streams[56] and programming with attributed state machines[34]. Among these issues, research most relevant to this thesis are synchronization and localization, both of which are surveyed in the following paragraph. However, interesting readers should refer to [10][13] for more general detail.

Every node has to be synchronized to either local or global time. Synchronization is crucial for applications that require fine time measurement, such as sensor data fusion, coordinated actuation, and power efficient duty cycling. Some localization algorithms use acoustic or RF time-of-arrival (ToA) or time-difference-of-arrival (TDoA) schemes to estimate the distance, and these schemes often require synchronization down to micro-seconds or even nano-seconds for fine accuracy. Synchronization has been attempted by ordering events[39], transmitting synchronization messages to the entire network[51] or only a set of neighbors [18] for example.

In an effort to study sensor networks in indoor environments, the Responsive Environment Group at the MIT Media Laboratory decided to build PLUG, a power

strip equipped with sensing, computational, and communication capabilities to form an IS that is natural to deploy in habituated environments. Using a set of PLUGs, the author developed a system for autonomous indoor localization and designed a simple environments drawing algorithm to display the PLUG's sensory data.

1.3.1 PLUG

As much as sensor networks have been studied for many years, discoveries have been confined only to the laboratory or specific tasks. They have not been "ubiquitous" enough to evolve into common applications. Accordingly, it is critical to actually deploy such sensors in a city. In order to be sustainably deployed, the system needs to be useful to people. The author and his colleagues believe that the sensors commonly embedded into our electronic appliances will form a ubiquitous network once common wireless and network standards are adopted. This infiltration into common objects will someday bootstrap a pervasive sensor network. Everyone carries around their cellular phones because they enable people to call other people almost anywhere anytime. Everyone also has power strips to power a large number of electronic devices simultaneously. These two electronic appliances are perfect examples that are useful to people, but also have enough resources to perform collection, processing, transmission and reception of sensor data. Although cellular phones can be used as sensor nodes, such as has been tried at the Media Lab by Dalton [15] and elsewhere by many others, because it is relatively easy to hack into power strips and power strips have essentially infinite source of power and ubiquitously deployed as essential items in every office and many houses, the power strip has been chosen as our model. Accordingly, PLUG has been developed based on these ideas. More detail on the PLUG is discussed in Chapter 4.

1.3.2 Localization

Being one of the most popular applications for what could be loosely termed sensor networks, camera networks range from surveillance and responsive environments to

scientific remote monitoring. Although distributed arrays of cameras have been used for decades in these applications, all processing tended to be done centrally, not distributed across the network. In these applications, network data is only useful if the data origin is known: an image of a thief without any location info does not give any useful information for security purposes. Manually measuring the location and orientation of every camera becomes infeasible, especially when the number of installed cameras is large. This example is only one of many that make the localization algorithm one of the important issues to consider.

Localization is important not only in the surveillance camera setting but also in many other applications. This importance was recognized by U.S. Government. In 1996, the US Federal Communication Commission (FCC) [22] required all wireless service providers to give location information to Emergency 911 services, called enhanced 911 or E911. By October 2001, the FCC mandates a $125m$ root mean square (RMS) accuracy for 67% of all 911 calls and by October 2006, a $300m$ RMS accuracy and 95% for all 911 calls.

Theoretically, many localization algorithms have been formulated as a decentralized detection problem, a term coined by Tsitsiklis, who formed a theoretical probabilistic foundation for this field. The model is beyond the scope of the thesis but interesting readers are referred to [81][32][85][64][84], where the sensor data is conditionally independent and to [86], which proves that the problem becomes NP-complete without the sensor data being conditionally independent. A survey of less theoretical but still interesting localization works is presented in Chapter 2 and Chapter 3. Some of these works have focused on active localization, meaning the localization is estimated based on signals that are artificially stimulated and measured by the sensor networks, such as transmitting RF or artificially generated acoustic events.

In addition, there is passive localization, which performs poorer but is more applicable to the real world. The basic idea behind passive localization is that, if different sensors sense a similar sound pattern, they determine that they are nearby each other with high probability; by comparing the timing of commonly detected transients, a localization can be executed. This problem has been studied in pervasive, for example

locating the position of gunshots with a localized sensor network [41]. Whereas active localization is performed in controlled environments, passive localization occurs in non-controlled environments where stimuli are autonomously generated. However, to author's best knowledge, no work has focused on passive localization by, for example, sounds of doors banging or foot stepping, nor on the combination of active and passive localization. Hence the author focuses on passive localization as well as the comparison between active and passive localization in a similar setting. In addition, the author also compares active acoustic localization with localization solely based on radio signal strength. These algorithms are discussed in Chapter 5 with corresponding comparison, both in simulation and experimental setup, in Chapter 6. The thesis is concluded in Chapter 7 with possible future studies.

Chapter 2

Background

Recent and evolving research in ubiquitous computing is reviewed in this chapter, with a particular concentration on autonomous localization algorithms. A goal of ubiquitous computing research is to deduce a global interpretation from distributedly-collected sensor data. The degree of the distributedness varies between projects and can be broken into four main categories.

1) First, network algorithms with no distributedness at all. In this algorithm, every task is centralized. One node is powerful enough to sense the environment, interpret the data, and display the results. These include, for example, satellite systems, where each node can be powerful enough to handle the entire data collection, processing and transmission by itself.

2) Second, network algorithms with low-level distributedness. In these works, each node is responsible only for collecting sensory data, filtering out the unnecessary data, and routing the filtered data to the base station for further processing. These algorithms are suitable for sensor network applications, where each node is not capable of heavy-computing nor heavy-storage. However, there are two drawbacks with these algorithms. First, they can cause network traffic, because entire network load is routed toward the base station. This network traffic might cause data loss, which can possibly lead to the faulty data interpretation. The second problem is that this algorithm is sensitive to outliers. Since there is no communication between nodes, outliers, which are usually due to noise and have no physical significance, might make

the node believe that they are interesting enough to be routed to the base station.

3) Third, network algorithms with mid-level distributedness. The only difference between second and third algorithms is that here each node filters out "intelligently" unnecessary data and outliers by talking to its neighbors and makes a local interpretation. This can reduce the network burden. Because different sensor data from adjacent nodes often carry redundant information, communication between adjacent neighbors can eliminate the redundancy. This algorithm is more scalable than other algorithms mentioned before. Some people argue that this would require each node to have more powerful processing power. However, if we think about this issue more carefully, it is not always true. Although there are two "additional" tasks, which are different from the second algorithm (communication between neighbors and simple local decisions), the author believes that these "additional" tasks are not truthfully additional. Communication between neighbors is already done in the second algorithm. Since every communication in the sensor network is done over multiple hops, this is already implemented in the second algorithm and requires no extra horsepower. Simple local decisions are made based on the data received from the node's neighbors. Although the neighborhood size depends on the network density, it is usually much smaller than number of sensor samples taken, indicating that each processor is capable of handling the data from its neighbors and making simple decisions and calculating simple parameters such as max, min, $median$, $average$, or $stdev$.

4) Last, network algorithms with high-level distributedness. These algorithms do not have any central aspect. Every interpretation is done locally. Although these are truly distributed algorithms and a perfect fit for sensor networks, most results on these algorithms yield either insufficient accuracy or require an extensive calibration period before the algorithm executes. Although such algorithms pose interesting challenges to solve, the author believes that it is not worth sweating over them from the sensor network application perspective for the reasons explained in the following paragraph. Please note that four categories mentioned will be referred to as **first**, **second**, **third**, and **fourth** categories throughout the Chapter.

Ubiquitous computing is a user-oriented paradigm. Thus, there is always a user

assumed with reasonably high processing power, waiting to encounter the results. The **third** class of algorithm is distributed enough to take advantage of the ubiquitousness of the sensors. At the same time, it is central enough to give the user an accurate interpretation and control over the sensor network. The author believes that the **third** types of algorithm is the most suitable approach for ubiquitous computing, with which the author's thesis work perfectly fits. More detail on the author's network algorithm is discussed in Chapter 5.

2.1 Localization Problem

For those readers who are not familiar with the localization issue, let us formulate a simple localization problem in a 1D plane with a perfect noiseless channel without any obstructions. Ten people form a straight line and each person can only transmit to his/her both right and left adjacent neighbor, if they exist. Each person would like to know how far he/she is from two ends of the line by only talking to their adjacent neighbors. The person without a left neighbor is indexed 1. Person 1 transmits signal to its right adjacent neighbor with unit strength. The person who receives from person 1 is indexed 2 and estimates its distance to person 1 based on the radio signal strength (RSS), which decreases typically with roughly the second (outdoor) and the fourth (indoor) power of distance [33][67]. Based on this information, person 2 estimates its relative location with respect to person 1. Person 2 transmits its location to its right neighbor, person 3; person 3 estimates its global location with respect to person 1. By propagating the location information down the line, every person can estimate their global locations with respect to person 1. This problem does not seem hard with given assumptions. However, once any one or any combination of these assumptions are relaxed, the problem gets much more complicated.

Most of the current localization algorithms are targeted for 2D or 3D domains. Being in a noiseless 2D domain introduces many issues that were not even considered in 1D domain due to many degrees of freedom such as reflection, translation, and rotation. It is worth noting here that some localization algorithms use "reference"

nodes to avoid some of the reflection, translation, and rotation problems. These "reference" nodes already know their location even before the localization algorithm executes, and will be referred as such throughout the thesis. In addition, the algorithm's complexity grows as optimum locations need to be estimated that meet the fine distance constraints. Noise models are usually modeled as Gaussian random noise in simulation, or raw noise data in real deployments. Formulating these via an approach termed SLAT (Simultaneous Localization and Tracking) has achieved recent popularity, originating in robotics studies, such as SLAM (Simultaneous Localization and Mapping).

2.2 SLAT(Simultaneous Localization and Tracking)

SLAT is an application of Bayesian inference, Kalman filtering, or other similar techniques to update a distribution over localizations and mobile trajectories as measurements become available [82]. As SLAT has a root in SLAM, SLAT inherits many similar characteristics from SLAM. SLAM algorithms help a robot to localize itself within a map of the environment, while concurrently building a map of its surroundings. As opposed to SLAM, SLAT is built around sensor network applications, hence has similar but different characteristics. Because robots are usually capable of carrying more powerful processors and massive storage than those typically used with sensor networks, their algorithms are not restricted to resource limitations. While the robot is moving around, the position of the robot and the angle of the radio, speaker, camera, or microphone can be estimated using data from various attached sensors and actuators. However, what can be challenging are the multiple objects to detect. Because the environment consists of multiple objects, both static and mobile, the navigating robot needs to be able to distinguish between these different objects in order to build its map. On the contrary, in the SLAT problem, the algorithms always assume only few objects to detect. However, the issue here is that there are multiple "robots" or sensor nodes that are detecting common objects. These sensor nodes do not have any information about their location or the angle of the onboard radio,

speaker, camera or microphone. Angle is one of many parameters that can affect the interpretation of sensor data. For example, distance measurement using radio signal strength (RSS) depends immensely on whether the transmitter and receiver are on a simple line-of-sight or not and also can depend on the relative angular position of the transmitter and receiver. This explains why most SLAT algorithms using RSS [3][25] include measurements of the angles of the transceivers.

What is lacking in a SLAM algorithm to become a SLAT algorithm is distributedness, scalability, and resource management. Many distributed algorithms specific for sensor networks are developed to solve these issues and two notable efforts are worth mentioning in detail with careful comparison to the author's work; this comparison appears in Section 2.4.3 and 2.4.4. First, let us survey the numerous other examples of localization algorithms, which form the foundation of the field. A comprehensive survey of localization systems can be found in [30] and a recent survey of localization algorithms in Savvides[71].

2.3 Range-Free Localization

Many sensor networks are constrained by resource limitations such as network bandwidth, processing power, memory, and cost of hardware. For resource limitations and many other reasons, range might be immeasurable in some cases but still these nodes require coarse localization. This field of study is called "range-free localization." Based on the number of RF beacon signals it received, each unlocalized node estimates its location using a simple centroid model, sometime referred to as the *Centroid Algorithm*[9]. Based on the radio network's hop count from the reference and the location of reference, the nodes calculate their position[54] [58]. Using the reference's location, an area-based range-free localization scheme, called APIT [29], estimates the node location by calculating the diameter of the estimated area in which a node resides. The APIT scheme performs best when irregular radio patterns and random node placement are considered, and low communication overhead is desired. Although the problem itself is interesting, most of the algorithms require the sensors

to be densely distributed in order to obtain a reasonable accuracy. Also, this algorithm is not robust against node or communication failures. The number of hops is not necessarily proportional to the distance and is sensitive to noise and outliers. For his thesis work, the author decided to rule range-free algorithms out from consideration in his localization system.

2.4 Range Localization

On the contrary, range localization requires each sensor node to be equipped with measuring devices such as an ultrasound transceiver, radio transceiver, microphone, etc. to estimate pair-wise distance. The research in range localization can be divided into two categories, depending on whether it is used indoors or outdoors. Outdoor range localization systems have been developed to the necessary accuracy thanks to the Global Positioning System (GPS) or the Galileo system, the European counterpart of GPS. The first Galileo satellite [53] was launched in 2005 and is expected to be operational by 2008 and completed by 2010 with 30 satellites. Its accuracy is projected to be within 3 feet, which is better than that of GPS, which has an accuracy of roughly 16 feet. Although there have been many successes for outdoor localization, there has been no such luck with indoor localization, despite much effort. Let us survey the current outdoor localization systems, and currently evolving indoor localization systems.

2.4.1 History of Outdoor Localization System

In 1970s, automatic vehicle location (AVL) systems were used to estimate the position of police cars and military ground transportation in urban settings [87]. A set of stationary base stations acted as reference points to estimate the pair-wise distances employing ToA (time-of-arrival) and TDoA (time-difference-of-arrival) schemes. Then, the vehicle position was estimated through multi-lateration, using Taylor series expansions [24] to transform a non-linear least squares problem to a linear one. LORAN(LOng RAnge Navigation)[95] was developed to provide outdoor terrestrial lo-

cation. LORAN estimates the location based on the time interval between radio signals received from three or more stations. Based on similar techniques, larger-scale localization schemes have been developed thanks to satellite systems. In 1996, the GPS (Global Positioning System) was developed to provide location information for civilian and military use. GPS [94] is satellite navigation system with two dozen GPS satellites broadcasting precise timing signals by radio to GPS receivers, allowing them to accurately determine their location, in longitude, latitude, and altitude using a TDoA scheme. Although the regular GPS system has accuracy of 16 feet, the Wide-Area Augmentation System (WAAS) and Differential GPS (DGPS) can increase the accuracy of GPS signals to within 6 feet and 3 feet respectively.

Another outdoor localization system is used for emergencies, exploiting terrestrial radio signal strength. For the E911 system [12], cellular base stations act as reference nodes to locate mobile phone telephone users within each cell. These schemes can be implemented in one of two ways: a mobile station uses signals transmitted by the base stations to calculate its own position, or the base stations measure the signals transmitted by the mobile station to estimate their locations. In addition, a network-based acoustic sniper localization system [41] is able to localize the position of a shooter and the trajectory of the projectile using observed acoustic events. The system is claimed to provide good coverage and high accuracy, being tolerant against multiple simultaneous acoustic sources, multipath effects, and multiple sensor failures. Another localization[78] algorithm makes use of additional knowledge of geometry deployment, where deployment is assumed to form a grid topology, achieving localization errors as low as 3% of radio range.

Early work at Lincoln Laboratory at MIT used differences of sound at several stations to passively track aircraft position [40]. A system developed at Vanderbilt University exploits the phase difference between beating RF carriers to localize an array of sensor nodes across a football field to *cm* scale accuracy [37], although multi-path effects will probably preclude indoor deployment. Stankovic's group also developed a system called Spotlight that used a hovering scanning laser range finder to localize sensor nodes scattered about a large open area [79]. Further survey of

radiolocation can be found in [12]. However, outdoor localization systems with high accuracy perform poorly in indoor environments, partially because of multipath effects. This calls for separate localization algorithms for indoor use.

2.4.2 Indoor Localization System

There are many localization algorithms using radios because 1) most radio transceiver packets provide radio signal strength indicator (RSSI) values and 2) although it is highly variable and depends on the environment, RSSI decreases with distance. An indoor location sensing system, called LANDMARC, [57] uses active RFID to localize. It has 4 RF readers, 16 tags as reference points and 8 tags to be tracked. All reference tags are organized in a grid array, and the localization uses the 4 nearest neighbors. Due to RFID's limited capability and the controlled environment that was required, LANDMARC did not show interesting results. The PicoRadio project at UC Berkeley [7] provides a geolocation scheme for an indoor environment, based on RF received signal strength measurements and calibrated signal strength maps, which require an accurate and extensive calibration period. PicoRadio is an ultra small wireless data transceiver node for sensor/actuator data. Based on the Electromagnetic Field Attenuation (EFA) map and sampled electromagnetic signal strength, it localizes each node [1]; the EFA map is generated for each base station, by starting from a set of acquired samples. During the operational phase, such maps are suitably intersected to compute the online position of the probe unit with a given attenuation signal. The EFA map from each reader is sent to the central station for localization purposes. 6 readers were used in MICA2 sensor networks; an area of 10.4 $\times$ 7.15 meters has been considered, giving 1.2 m error with 50% confidence.

RADAR [3], probably one of the most cited localization algorithm using RSS, presents two methods: an empirical method and method using a radio propagation model. Both approaches employ three reference points in an area of $43.5m$ by $22.5m$. The empirical method requires extensive calibration of the relation between distance and RSS at 70 equally distributed pre-known locations with four different orientations in each location. This calibration gives the basis to perform triangulization to track

either static or dynamic objects. The second method employs a radio propagation model. This technique did not use Rayleigh fading model or Rician distribution model due to their unrealistic assumptions. RADAR used a Floor Attenuation Factor propagation model, which provides flexibility in accommodating different building layouts, while taking into account large-scale path loss. Based on this model, it calculates distance. The first model performs with $2.94m$ error and the latter performs with $4.3m$ error. However, the second model is scalable and requires less resources than the first one. There is always a trade-off between the length of the calibration period and the algorithm accuracy. The more extensive the calibration is, the better the localization algorithm with calibration performs. However, more extensive calibration requires extensive memory use and time, which are not always possible with sensor networks. Whitehouse [93] suggests some ways to calibrate, which makes it easier for localization algorithms with a calibration phase, but still requires a comparatively large amount of memory and effort.

The Cricket location support system [66] provides an indoor location using an ultrasound signal for pair-wise distance estimation; the distance is estimated based on the timing of the acoustic signals relative to a trigger broadcast over a radio link. Based on the estimated pair-wise distance and fixed references with known locations, each unlocalized node receives a signal from different references in a randomized schedule to compute a maximum likelihood estimate of location with a granularity of 4 by 4 feet. As opposed to the Cricket system, where each non-reference node is a signal receiver, a BAT [28] [27] node transmits ultrasound signals, and these signals are picked up by an array of receivers mounted on the ceiling. The location of a BAT is estimated via multi-lateration with a few centimeters of accuracy. An RF base station coordinates the ultrasound transmissions such that interference is avoided. Although the Cricket and the BAT set a standard for localization using ultrasound signals, they require a centralized infrastructure and an "unnatural" signal (ultrasound) that requires extra measurement tools.

AHLoS (Ad-Hoc Localization System) [72] uses RF and ultrasound transmissions and does not rely on a preinstalled infrastructure. With a distributed localization

algorithm running at every node and a small fraction of the nodes aware of their locations beforehand, nodes can estimate their location even if they are not within the range with the reference nodes. In the EcoLocation algorithm [100], the unknown node location estimate is obtained by comparing the constraints obtained from RSS measurements to the constraint sets of each location grid-point and picking the location that satisfies the maximum number of constraints. There can be arbitrarily many reference and unknown points. This paper[100] presents the Ecolocation algorithm and its comparison with other localization algorithms. It is worth noting the following equation. The list of the parameters with appropriate values for indoor use is discussed in Section 5.6.1

$$RSS(d) = P_T - PL(d_0) - 10\eta \log_{10} \frac{d}{d_0} + X_\sigma$$

However, all these localization algorithms rely heavily on the calibration and exploit an unreliable distance measurement source. The timing of ultrasound signals and the radio signal strength depend not only on distance, but also on the structure of the obstructions, temperature, wind and many other factors. Because acoustic and RF measurements are sensitive to noise, despite the over-determined measurements, the localization result tends to be coarse. Elnahrawy [17] presents strong evidence that every localization algorithm using signal strength has the same fundamental limitations and these are unlikely to be overcome without more complex environmental models or additional localization infrastructure. The metric used were: average tile accuracy, average precision, average room accuracy, and average room precision. Below, the author would like to discuss two notable works that do not use ultrasound and radio as major source to estimate the distance. Because these works greatly influence author's thinking process, the author would like to discuss them in depth, comparing these with author's own work.

2.4.3 Networked Cameras

For the 2D localization problem with Gaussian noise, Funiak [25] presents an autonomous distributed solution by using information provided by an entire camera network. Every camera is assumed to be synchronized and it is assumed that there is only one moving object to detect. These assumptions can be sometimes impractical; there are usually more than one moving object in real indoor environments and it's hard to synchronize the entire network. Both of these two assumptions are relaxed in the author's thesis work to make the localization algorithm more applicable to a real world deployment. Nevertheless, these two assumptions greatly simplify a complicated problem.

Synchronized cameras are placed throughout the environment randomly at unknown locations. Then, as an object randomly moves around, the network automatically estimates the camera's pose, i.e., the combination of location and the orientation of the cameras. The `object_state` (See below for detail) needs to be estimated as well, because these two variables are inter-dependent with each other. The use of cameras in localization is hardware costly and introduces extra variable to measure, such as camera's orientation. This motivates the author to use omni-directional sensors, measuring light, sound, and vibration, in the localization process in order to eliminate an additional variable at low cost.

For scalability purposes, Funiak uses distributed probabilistic inference on network junction tree data structures, which were introduced by Paskin[62]. A distributed algorithm eventually needs to flood the entire network with necessary messages to achieve accurate performance, and this usually requires exponential growth of resource use as the number of nodes increases. In order to avoid this exponential growth, rather than representing the belief state as a monolithic probability distribution over all state variables, he uses an approximation. In other words, instead of flooding the entire network with information, nodes transmit data to only subset of the network to process based on a network junction tree data structure. Intuitively, to avoid the cost of maintaining dependency information between all variables, dependencies are

instead maintained between small, overlapping subsets of variables. See Appendix A for more technical details.

Funiak's work is designed for updatability and scalability purposes. Funiak tests this algorithm in simulation for both the **third** category and the **fourth** category of networking algorithms. As expected, the **third** category performs with smaller error because every node has the common `belief_state`. Although it is easy in simulation, letting every node have the common `belief_state` can be cumbersome in real deployment. One popular way this can be accomplished is to have the base station periodically broadcast the `belief_state` to the entire network, but this might cause network traffic congestion and data packet loss.

For updatability purposes, the author instead decided to use a simple average for both pair-wise distance and coordinate estimation. The author believes that this greatly reduces the memory usage, saves processing power, and performs well in comparison to the network junction tree. For scalability purposes, all communication is done over a multiple hop scheme. As opposed to cluster-forming algorithms, which might also depend on the network size, the author's algorithm filters/interprets data based on communication between one-hop neighboring nodes.

2.4.4 LaSLAT in Ad Hoc Sensor Networks

Taylor [82] presents LaSLAT (Laplacian SLAT) as one of the solution to SLAT problem. Just like Networked Cameras, as explained in Section 2.4.3, a Bayesian filter uses distance measurement to the moving target to update a joint probability distribution over the positions of the nodes, the trajectory of the target and the calibration parameters of the network. Only the distance between the global stimulus and the sensors is measured, avoiding pair-wise distance measurement between sensors. Taylor claims that measurement noise is automatically averaged out as more measurements become available, improving localization and tracking accuracy in high-traffic areas. The filtering process is done on the local network, providing online estimates of all locations, calibration parameters and their uncertainties. There can be multiple targets, moving arbitrarily through the environment, with no constraint on their trajectory or

velocity. This algorithm was done in both a central and distributed way, lying in the **third** and **fourth** network categories respectively. Their illustration was performed on the Cricket nodes[76][66], which are capable of measuring their distance to a moving reference using a combination of ultrasound and radio pulses in the 2D domain. The algorithm was also tested in the 3D domain, but showed poor accuracy.

As always, localization algorithms with matrix operations have non-linear Gaussian problems. This non-linearity was handled by approximate linearization by Funiak, and Taylor handles this using Laplace's method. The technical detail here is beyond the scope of the thesis, but interested readers are referred to [82] for further development.

2.4.5 Affine Structure From Sound

Compared to the two aforementioned works, Thrun [83] presents a simple pair-wise distance measurement scheme and localization method based solely on acoustic events. He assumes that these acoustic events happen far away enough from the sensor array that the incoming sound wave hits the sensor array at approximately the same incident angle, named the *far field approximation*. The lines connecting the location of an acoustic event with each of the sound sensors are approximately parallel. Thus, the distance between two sound sensors is estimated based on the differential delay times. Using this scheme, Thrun measures the pair-wise distance. Based on this set of pair-wise distances, the least squares problem is solved to find the optimum coordinates for all sensor locations. As opposed to distance measuring tools using camera vision, ultrasound, or radio pulses, audio sound does not require any special equipment on the moving object's side and the sound sensor on the nodes are comparatively cheap. The author decided to use mainly sound in his localization algorithm. However, *far field approximation* aims for the 2D localization, thus the author decided not to use it in his implementation.

Chapter 3

Related Works: Basis of Author's work

Based on the extensive studies of localization outlined in Chapter 2, the author decided to use sound, vibration and light sensor values to estimate the pair-wise distance and not to use RSS values. The performance comparison between the localization using sound sensors and the localization using RSS values will be presented in Chapter 6. The localization problem, sometimes called the "sensor layout problem" [26], states as follows:

> Given a set of unlocalized nodes and a mechanism by which a node can estimate its distance to its neighbors, determine the coordinates of every sensor via local communication.

To author's best knowledge, many localization algorithms solve the sensor layout problem by using one of following techniques or variations of them: lateration algorithms, least-square estimators, Kalman-filtering, mesh-relaxation[31], or spectral graph drawing. Based on measured distances, the locations are estimated up to a rotation, translation, and reflection. However, localization sometimes requires precise global location with accurate rotation, translation, and reflection, and this mandates the use of reference nodes. Reference nodes know their positions, even before the algorithm begins, and other nodes estimate their locations based on that of the reference

nodes. Although some attempts [65] have been made to avoid the use of reference-based algorithms, the author believes that this work was not sufficiently tested in the field to be applicable to author's work. Hence, the author decided to use two reference nodes in his approach. Many schemes used in the author's pair-wise distance estimation, as presented in Section 5.6, and the localization algorithms are influenced by the two localization schemes mentioned in the remainder of this chapter.

3.1 Audio-Based Localization For Ubiquitous Sensor Networks

Ben Dalton [15] at the MIT Media Lab presented an active acoustic source location estimation method for microphone resources of network-connected heterogeneous devices containing distributed processors and uncalibrated sensors. His method used a least-square estimator to converge to the true positions and was tested on the Smart Architectural Surfaces (SAS) development platform in a mixed-device ad-hoc sensor network. Although this work shows great potential to localize over heterogeneous devices, these devices are capable of heavy computation, which is not always possible. His algorithm thus did not consider any resource, memory, or processing power limitations. He used only the timing of sound pulses and assumed that every node is synchronized. From this work, the author decided to use sound as one basic measurement to estimate the pair-wise distance. However, the author used a matched filter algorithm instead of pair-wise ranging to calculate the time delay. Also, the author used an average-lateration scheme, discussed in Section 5.7, instead of a least-square's estimator.

3.1.1 Smart Architectural Surfaces(SAS)

As a brief diversion, SAS[74] are modeled as highly integrated and interactive "smart spaces" based on a self-organizing network of cells. Each cell is capable of networking, sensing "intelligently" and actuating/displaying. (See Figure 3-1 for illustration.)

Figure 3-1: Smart Architectural Surfaces tiles mounted on a frame along two glass walls. A distributed data display can be seen across the screens of one wall. This picture is excerpted from Dalton's thesis [15].

The SAS was made to be a unobtrusive, physically and logically re-configurable environment. Each tile can operate in isolation, but benefits from connection and collaboration, favoring the idea of viral networking[46] and the ultimate goal of sensor networks. This project has a similar goal as PLUG, presented in Chapter 4, in its unobtrusiveness and viralness.

3.2 Localization and Sensing Applications in the Pushpin Computing Network

Michael Broxton presented two systems for localizing a network of roughly 60 pushpin nodes distributed (shown in Figure 3-2) over an area of $1m^2$[8]. The first was based on a linear lateration technique, while the second approach utilized non-linear optimization techniques, namely spectral graph drawing and mesh relaxation. Pair-wise distances are estimated based on ultrasound time-of-flight measurements from three global opto-acoustic sensor stimuli. With this method, a localization error of $2.30cm$ and an error standard deviation of $2.36cm$ were achieved.

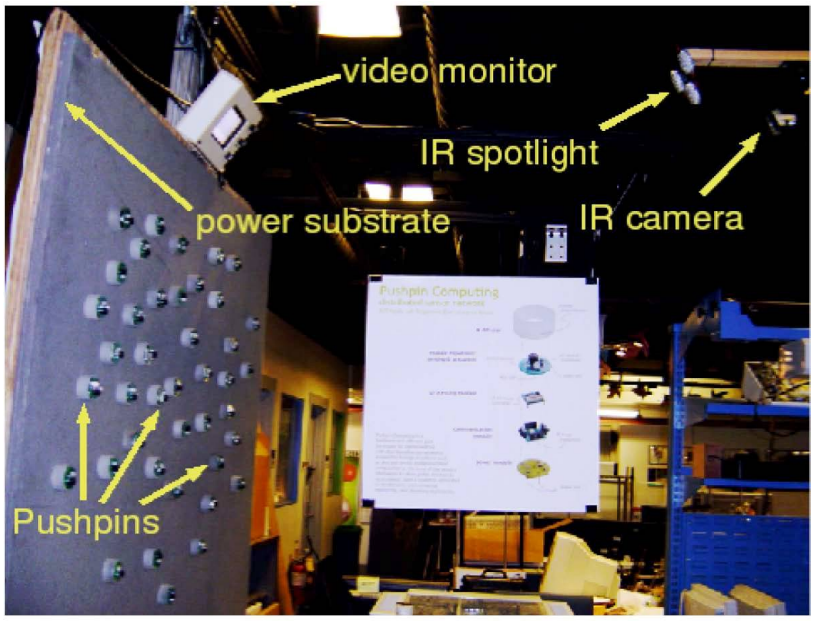


Figure 3-2: Pushpin experimental setup. This includes an IR spotlight for synchronization and parallel programming. An IR sensitive camera and video monitor are used to display the IR communication path. This picture is excerpted from Broxton's thesis [8].

3.2.1 Linear Algorithm: Lateration

Given the positions of and distances to the reference nodes, Broxton forms a system of equations for the localization problem in 3D:

$$\begin{bmatrix} 2(x_1 - x_4) & 2(y_1 - y_4) & 2(z_1 - z_4) \\ 2(x_2 - x_4) & 2(y_2 - y_4) & 2(z_2 - z_4) \\ 2(x_3 - x_4) & 2(y_3 - y_4) & 2(z_3 - z_4) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1^2 - x_4^2 + y_1^2 - y_4^2 + z_1^2 - z_4^2 + d_4^2 - d_1^2 \\ x_2^2 - x_4^2 + y_2^2 - y_4^2 + z_2^2 - z_4^2 + d_4^2 - d_2^2 \\ x_3^2 - x_4^2 + y_1^3 - y_4^2 + z_3^2 - z_4^2 + d_4^2 - d_3^2 \end{bmatrix}$$

where (x, y, z) is the location of the nodes to localize, (x_i, y_i, z_i) for $i = 1, 2, 3, 4$ is the location for reference i respectively and d_i is estimated distance between nodes to localize and reference i. With more reference nodes, a linear least squares estimate can be used. The author uses a similar technique, but in a different format due to the unavailability of global stimuli that are experienced by all nodes.

3.2.2 Non-Linear Algorithm: Spectral Graph Drawing (SGD)

SGD constructs the layout using eigenvectors of certain matrices associated with the graph. This approach originated in 1970 by Hall [35], but was not used much until it was picked up by Koren [36]. Koren applied the spectral graph drawing scheme to solve the sensor layout problem [26].

However, this problem does not necessarily have a unique solution. When all $\binom{n}{2}$ pair-wise distances are known without measurement error, the solution is unique; otherwise, a more sophisticated solution is necessary: SGD. In order to solve this problem, two main steps are performed. First, an initialization process, where the 1D localization problem is formulated as follows [26]:

$$\vec{x} = \arg_{\vec{x}} \min \left[\frac{\sum_{<i,j>\in E} w_{i,j} ||x_i - x_j||^2}{\sum_{i<j} ||x_i - x_j||^2} \right]$$

where E is the set of connected edges, $\vec{x} = \{x_1, x_2, ..., x_n\}$ and x_i is a x coordinate of ith node. This function makes intuitive sense because it tries to locate adjacent nodes close to each other while separating nonadjacent nodes. The same idea has been applied to formulate the 2D localization problem. After careful manipulation, $\vec{x}$ turns out to be the v_2 eigenvector of $D^{-1}W$ and $\vec{y}$ is v_3 eigenvector of $D^{-1}W$. Here, D is an $n \times n$ diagonal matrix with $D_{ii} = \deg(i)$ and W is $n \times n$ matrix where w_{ij} is the weight between node i and node j, where weight increases with the estimated distance. v_2 and v_3 are the eigenvectors corresponding to the second and third largest eigenvalues of $D^{-1}W$. These eigenvectors are found by a power iteration method. Once the initial locations are estimated, an update process is applied. Every node's location becomes the weighted centroid of its neighbors. However, this technique requires an extensive set of pair-wise distances to be measured, which are not always available in the author's setting, where distance estimation between two nodes is possible only when a sonic event near these nodes is audible to both nodes. Therefore, the author decided not to use this technique.

Chapter 4

PLUGs

In May 2005, a workshop was held at the Pervasive Computing Conference in Munich, Germany for the following purpose[97]:

> The majority of application studies presented at conferences in sensor network research are exemplars of potential applications suggesting directions for further research. But they often contribute little to our understanding of the broader needs of users of ubiquitous systems and the wider potential of the underlying technologies. Nor do they provide a context within which the merits of alternative designs can be effectively assessed. Ubicomp research could benefit from a better-mapped domain for application research with established metrics, methods for the selection, analysis and evaluation of applications and common infrastructures.

As pointed out in the above paragraph, sensor network research needs to be application-driven. As much as we learned from many lessons with our previous sensor network platforms such as Pushpin Computing (Chapter 3), we realized that these projects do not tell us too much besides how they perform in a controlled environment. PLUGs have been developed to meet application driven goals and to focus on new algorithms such as routing, data fusion, localization, etc, in an indoor setting instead of focusing on the hardware issues and power efficient algorithms. Because in most indoor settings, PLUG can have an access to infinite power source, the author and his colleagues

assume that power is not so much of an issue here. As sensor networks for ubiquitous computing are expected to merge into everyday electronic appliances, the PLUGs can be unnoticeable as they are camouflaged in power strip while occupying a minimum size. The author would like to point out that most of the hardware design and development were done by his colleagues in the Media Lab's Responsive Environments Group, Mark Feldmeier and Josh Lifton. Its OS and basic applications have been developed mainly by Josh Lifton and has been debugged and tested by Josh Lifton, Yasuhiro Ono, Bo Morgan, and the author.

4.1 Motivation

The Kansei testbed [19] at the Ohio State University is designed to facilitate research on networked sensing applications at scale. This contains a set of heterogeneous nodes, each dedicated for local computation, storage, data exfiltration, and back-channel communication, to support complex experimentation. It also contains a real-time synchronized simulation engine for sensor data display. See Figure 4-1 for illustration. The ORBIT Radio Grid Test bed (Open Access Research Testbed for Next-Generation Wireless Networks) [68], developed at Rutgers University's Wireless Information Network Lab, has been acclaimed for its usefulness as a wireless network test bed. It consists of an indoor radio grid emulator for controlled experimentation and an outdoor field trial network for end-user evaluations in real-world settings. As a wireless network protocol, it is used for different stages of protocol design, evaluation and testing. In addition, there are numerous other wireless test beds[44], which are built for similar purposes. However, these test beds are used only by knowledgeable researchers and often for artificial purposes. The author believes that the technical issues sometimes can be easily solved by considering user experiments with non-technical people rather than exclusive use by technical people, as in the software engineering domain, where software does not become available to public until its beta version has been fully tested by non-technical people. Accordingly, the PLUGs are developed as a "beta version" to test out existing algorithms and devise new ones as

Figure 4-1: Kansei is a testbed of 210 Extreme Scale Motes (XSM) hooked individually onto 210 Extreme Scale Stargates (XSS). This provides a test bed infrastructure to conduct experiments with 802.11b networking and XSMs. The picture is taken from Kansei's homepage `http://ceti.cse.ohio-state.edu/kansei/`

it becomes necessary.

4.2 PLUG Hardware

PLUG contains two parts: a low voltage part and a high voltage part. The low voltage part contains an AT91SAM7S64 microcontroller, a CC2500 transceiver and sensors. (See Table 4.1 for detail). The high voltage part mainly acts as a programmable current provider to each outlet and to the low voltage board. In addition, it includes fuses and safety devices.

4.2.1 Low Voltage Board

The low voltage board contains two main parts: the AT91SAM7S64 and the CC2500.

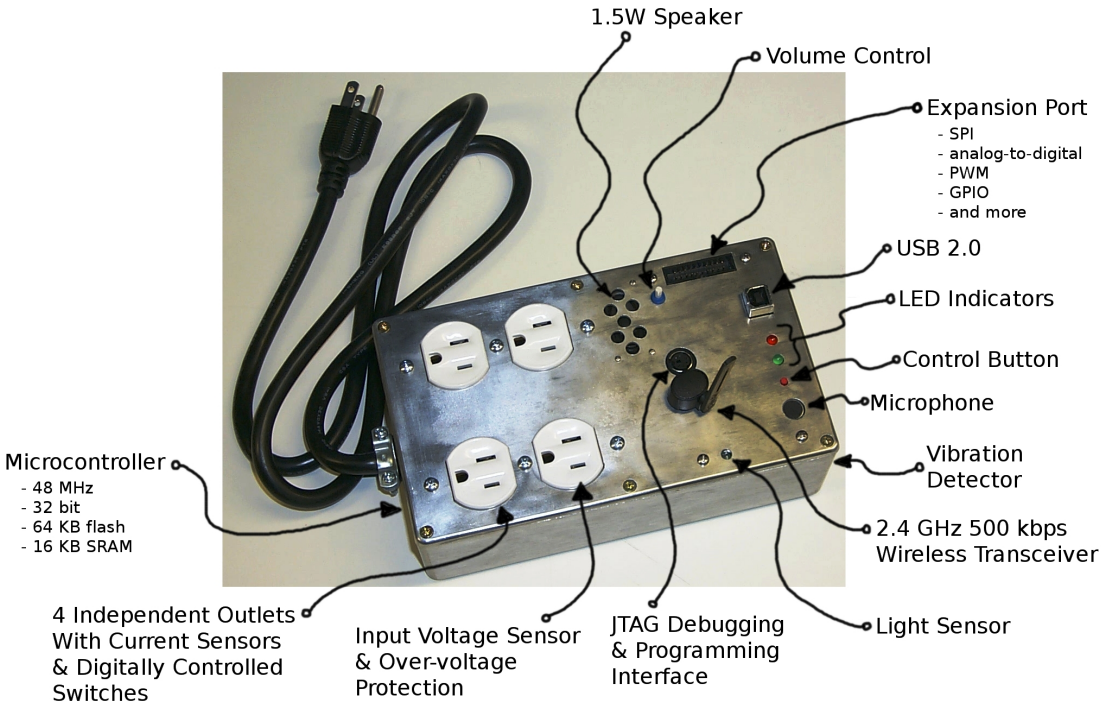


Figure 4-2: PLUG with annotation

Parts
Atmel ARM7-based AT91SAM7S64 microcontroller
Analog Devices SSM2211 low distortion 1.5 Watt audio power amplifier
knob to control the volume of the speaker
JTAG interface for programming and debugging the microcontroller
USB connector
Two LEDs
phototransistor sensor
audio microphone
vibration sensor
Chipcon CC2500 2.4 GHz RF Transceiver

Table 4.1: Important Parts in the Low-Voltage Board for PLUG

AT91SAM7S64 microprocessor

The AT91SAM7S64 [2]is a low pincount Flash microcontroller based on the 32-bit ARM7TDMI RISC processor. It features 64K bytes of embedded high-speed Flash with sector lock capabilities and a security bit, and 16K bytes of SRAM. The integrated proprietary SAM-BA Boot Assistant enables in-system programming of the embedded Flash. Its extensive peripheral set includes a USB 2.0 Full Speed Device Port, USARTs, SPI, SSC, TWI and an 8-channel 10-bit ADC. Its Peripheral DMA Controller channels eliminate processor bottlenecks during peripheral-to-memory transfers. Its System Controller manages interrupts, clocks, power, time, debug and reset, significantly reducing the external chip count and minimizing power consumption.

CC2500

The CC2500 [14] is an integrated multi-channel RF transceiver designed for low-power (13.3mA in RX, 250 kbps, input 30 dB above sensitivity limit) wireless applications in the industrial Scientific Medical (ISM) band at 2400-2483.5 MHz. It contains a separate buffer for 64 bytes received and transmitted data in FIFO fashion. It also provides a digital radio signal strength indicator (RSSI) and digital link quality indicator (LQI). Use of the RSSI in localization is further discussed in Chapter 5 and Chapter 6.

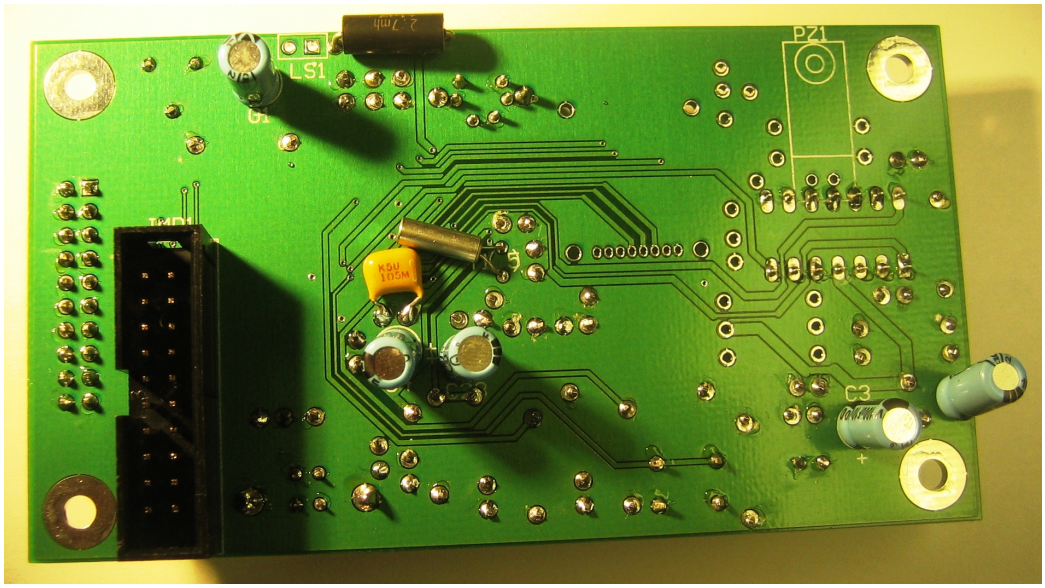


Figure 4-3: Low-Voltage Part for PLUG

4.2.2 High Voltage Board

This board contains a set of Fairchild Semiconductor MOC3023 6-Pin DIP 400V Random Phase Triac Driver Output Optocouplers [21], an optically isolated triac driver device, to control the outage current provided by each outlet. Each outlet is equipped with Triad CSE-1871 current sensor transformers to measure the AC current. Thermal heat sink paste is used to better conduct heat between these parts and their heat sinks. In addition, the high voltage board contains a 20-pin connector for mating the high voltage board to the low voltage board.

4.3 Software

Due to memory constraints, the operation system and application modules, such as the hardware testing unit and localization unit, are combined into one package and uploaded onto RAM. For explanation purposes, the author makes a distinction between localization-related software and everything else, and the localization-related software is explained later in Chapter 5. *Everything else* can be divided into two parts: the OS and the interface from PC side. In this section, several libraries that constitute

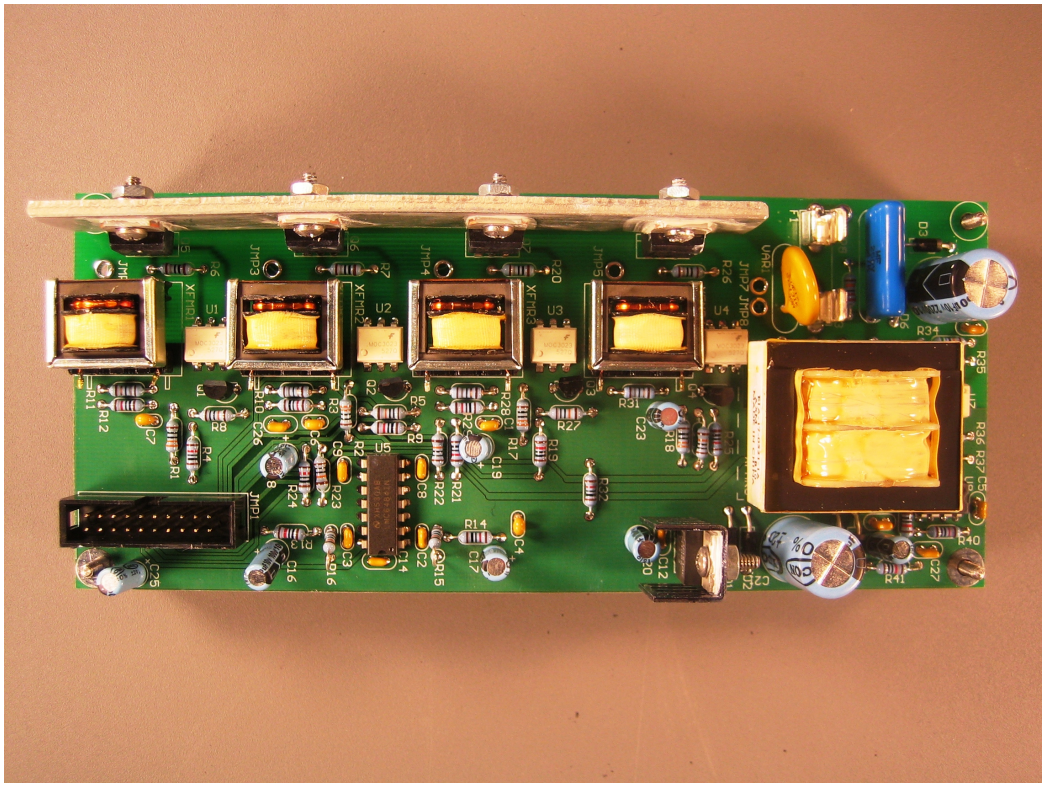


Figure 4-4: High-Voltage Part for PLUG

the OS are presented along with the PLUG interface.

4.3.1 Low-Level OS

Software is written in the ANSI-standard C. The C Library is written to handle hardware in different layers.

4.3.2 Basic Module

1. `AT91SAM7S64.h`: This file was provided by Rowley Associates Limited, who provided the C compiler that we used. It has wrappers for every component of AT91SAM7S64 with reasonable mnemonics. This file forms a basis to control peripherals, such as sensors, the CC2500 transceiver, and outlet current control, which are all attached to the microcontroller.

2. `types.h`: This file defines data types to be used. It includes both signed and unsigned numbers in 8 bit, 16 bit, 32 bit and 64 bits.

3. `system.c and system.h`: This defines interrupt tags (see Table 4.2 for detail), master clock frequency (47923200 Hz), RC oscillator frequency (32768 Hz) and crystal frequency (18432000 Hz). These frequencies form a basis for the interrupt handler and clock.c and clock.h. For fast system access, `system_init` and `system_reset` are defined. `system_init` initializes the basic features of the microprocessor, including the watchdog timer, spurious interrupt handler, master clock, and hardware interrupt vectoring. In addition, it defines `system_error` for easy debugging.

4. `clock.c and clock.h`: The system uses the alarm data structure to create and manage an arbitrary number of alarm clocks. Each instance of alarm represents a single task to be performed at a designated time. Although each task occurs accurately at the designated time, they will encounter delay when multiple tasks are assigned at the same time. More accurately, multiple tasks assigned at the same time are handled in the order registered, but with about 15 μs of delay between two consecutive tasks. Necessary conventional time clocks are also defined. One second corresponds to 128 Master Clock ticks, making x seconds to be $47923200x/128$ master clock ticks. For example, `CLOCK_1_DAY` is defined as $47923200 \times 60 \times 60 \times 24/128 = 32348160000$. To implement these, `clock_get_time` is defined to return the time, `clock_set_alarm` to set a certain task at a certain time, `clock_init` to initialize the time, and `delay` to delay for an arbitrary number of master clock ticks. `delay` is needed sometimes before transmitting data wirelessly to avoid network congestion.

5. `pioa.c and pioa.h`: This file enables the parallel I/O Controller A. This contains the definition for the PIOA device descriptor structure and functions to initialize and register the PIO device and PIOA interrupt handler.

6. `adc.c and adc.h`: This module converts analog sensor data to 10 bit digital data. This module defines two functions. The first function, the bit stuffer, maintains an efficient array of bytes that are stuffed with 10-bit ADC values by overlapping parts of different 10-bit values within the same byte instead of considering the 10-bit ADC value as a 16-bit integer. The second function starts the analog to digital converter (ADC) according to the parameters passed as arguments. Parameters include information regarding 1) the number of samples per second, 2) the total number of bits per sample, 3) enabled channels, 4) number of bits per sample, and 5) the implementation when the sampling is done. In addition, this file contains the ADC interrupt handler.

7. `spi.c and spi.h`: This file enables the serial peripheral interface. This contains a definition for the SPI device descriptor structure and four functions to initialize, register the SPI, set the current SPI device, and indicate whether it is busy or not.

8. `button.c and button.h`: PLUG contains one button to take input from the user. This is defined as one instance of the parallel input/output device (`PIOA`). This function defines a structure for Button and the structure contains two pointers to the events when the button is released and depressed respectively. Also, it contains an initialization function. This button is used as a simple way for user to provide input.

9. `cc2500.c and cc2500.h`: This file provides an interface for the ChipCon CC2500 2.4-GHz transceiver. This forms a basis for network.c and network.h to transmit and receive data. The transceiver is defined as serial peripheral interface (SPI) as opposed to parallel input/output device (PIOA) for optimum performance. This file provides transceiver initialization (`cc2500_init`), reception initiation (`cc2500_initiate_receive`) and transmission initiation (`cc2500_transmit_initiate`) to send any buffered packet in FIFO fashion. In addition, it also defines the SPI interrupt handler.

Tags	Description
INTERRUPT_SYSIRQ	System Interrupt
INTERRUPT_PIOA	Parallel I/O Controller A
INTERRUPT_ADC	Analog to Digital Converter
INTERRUPT_SPI	Serial Peripheral Interface
INTERRUPT_US0	USART 0
INTERRUPT_US1	USART 1
INTERRUPT_SSC	Synchronous Serial Controller
INTERRUPT_TWI	Two-wire interface
INTERRUPT_PWMC	PWM Controller
INTERRUPT_UDP	USB Device Port
INTERRUPT_TC0	Timer/Counter 0
INTERRUPT_TC1	Timer/Counter 1
INTERRUPT_TC2	Timer/Counter 2

Table 4.2: Interrupt Tags for PLUG defined in system.h /citedatasheetAT

10. `leds.c and leds.h`: This file provides ways to toggle, blink and turn on/off both red and green LEDs. These LEDs are used to let the user know of arbitrary program status.

11. `speaker.c and speaker.h`: This initializes the peripherals necessary to control an attached speaker using the pulse width modulation (PWM) controller. This file defines a structure for `Sound_t` and `Speaker_t`, both of which are used to make sound. In addition, it defines a PWM interrupt handler to play audio through the speaker.

12. `switches.c and switches.h`: This file provides ways to turn on/off the current to each outlet. It initializes each outlet with "on."

13. `usb.c and usb.h`: This file provides an interface to transmit/receive data from the microcontroller over the USB 2.0 port. This file contains functions to initialize, read/write data, and send null packets. In addition, it also contains a USB interrupt handler.

4.3.3 High-Level OS

1. `random.c and random.h`: This contains a function for a random number generator for both 16 bit and 32 bit values. This function uses an instantaneously detected sound and light sensor values and a standard signed random generator.

2. `network.c and network.h`: This defines the function for packet transmission and reception and initialization. In addition, this file defines the structures for networking, namely `Packet`, `Neighbor` and `Network`. `Packet` contains payload length, destination address, starting address, type of the packet and payload content. The network packet type is listed in Table 4.3 with its description. Each node is allowed to have a maximum of 16 neighbors and keeps track of each neighbor's information, such as local address, link quality (provided by CC2500), radio signal strength indicator (provided by CC2500), packets received, time when the last packet has been received, and its global address. The data structure of `Network` contains a pointer to the most recently received packet, its local address and the global address. The node's local address is generated randomly every time it initializes; although it is unique among its neighbors, it is not necessarily unique in its entire network. However, its global address is unique in its entire network and it does not change.

3. `gradient.c and gradient.h`: This creates, removes, and refreshes the gradient. This file provides functions to initialize, create, refresh and remove the gradient and update the gradient table. A gradient is created when the source is triggered and it acts as a communication path from the source to all the other nodes within a specified maximum number of hops. This specified maximum number of hops can possibly form a cluster inside a network so that the "infection" does not affect other clusters.

4. `sounds.c and sounds.h`: This function defines what we term a chirp sound, ramp sound and noise burst sound. This function is used to generate the natural sounds in the author's algorithm.

Tags	Description
PACKET_BROADCAST	send to all neighbors once
PACKET_PING	send to a specific neighbor
PACKET_REQUEST_TO_SEND	beginning of an acknowledged packet transaction
PACKET_CLEAR_TO_SEND	recipient's response to request to send if clear
PACKET_ACKNOWLEDGE	recipient's response to successful acknowledged packet receipt
PACKET_FAILED	recipient's response if not clear to send or packet didn't arrive
PACKET_NETWORK_BROADCAST	broadcast to all nodes in the network
PACKET_GRADIENT_BUILDER	builds gradient according to the specified maximum hop count limit.
PACKET_GRADIENT_REMOVER	removes a given gradient by removing the gradient routing table
PACKET_GRADIENT_CLIMBER	traces a given gradient back to its source by following a path routing table.
PACKET_GRADIENT_CLIMBER_BUILDER	traces a given gradient back to its source by following a path routing table. Along the way, it creates a new gradient along the path so that the original gradient source can reply through a direct path.

Table 4.3: Network packet type

5. `vibratab.c and vibratab.h`: The vibration sensor is defined as PIOA device. This file provides a structure for the vibration tab in addition to its initialization function.

6. `hardware_test.c and hardware_test.h`: Everytime the hardware is turned on, this function checks for the hardware soundness.

4.4 PLUG Interface from the PC side

This package is written in python and consists of three files.

1. `plug.py`: Under class `LocalPlug`, this defines functions to transmit necessary commands to the designated PLUG over the USB port. These commands include `createGradient`, `refreshGradient`, `startDataCollection`, `stopDataCollection`, `getID`, `requestNetworkPacket`, `printAllNetworkPackets`, `logNetworkPackets`, and `getNeighbors`. When any of these commands is called, certain constants are transmitted over the USB cable to the connected PLUG. Under the class `NetworkPacket`, this provides ways to parse the network packet and print it in readable way. As defined in the firmware, each network packet contains payload length, destination address, source address, message type and actual payload. In addition, it also contains RSSI, link quality indicator (LQI) and cyclic redundancy check (CRC), all of which are provided by CC2500 packet.

2. `plugutil.py`: This defines different functions to convert integers to various formats, necessary to process data.

3. `plugusb.py`: This is an interface to `usb.c` and `usb.h` in firmware. It defines functions to transmit/receive data from PLUG over USB 2.0. In addition, it provides a way to look for a device with particular vendor ID, product ID and interface ID among devices available on USB buses.

4.5 CrossWorks

The CrossWorks development tool is used to upload and debug the applications for the AT91SAM7S64 microprocessor. The CrossWorks for ARM is a complete C development system for microprocessors, consisting of the ARM GCC C compiler, the CrossWorks C Library and the CrossStudio integrated development environment (IDE)[45]. Its C Library has been redesigned for specific use within embedded systems conforming to the ANSI and ISO C standard. Key features of the IDE are debugging tools in addition to the source code editor, project organizer, and build system. An ARM Hardware Debugging tool lets you use the integrated debugger to step through the software on the target board. An ARM Flash Programming and Debugging tool lets

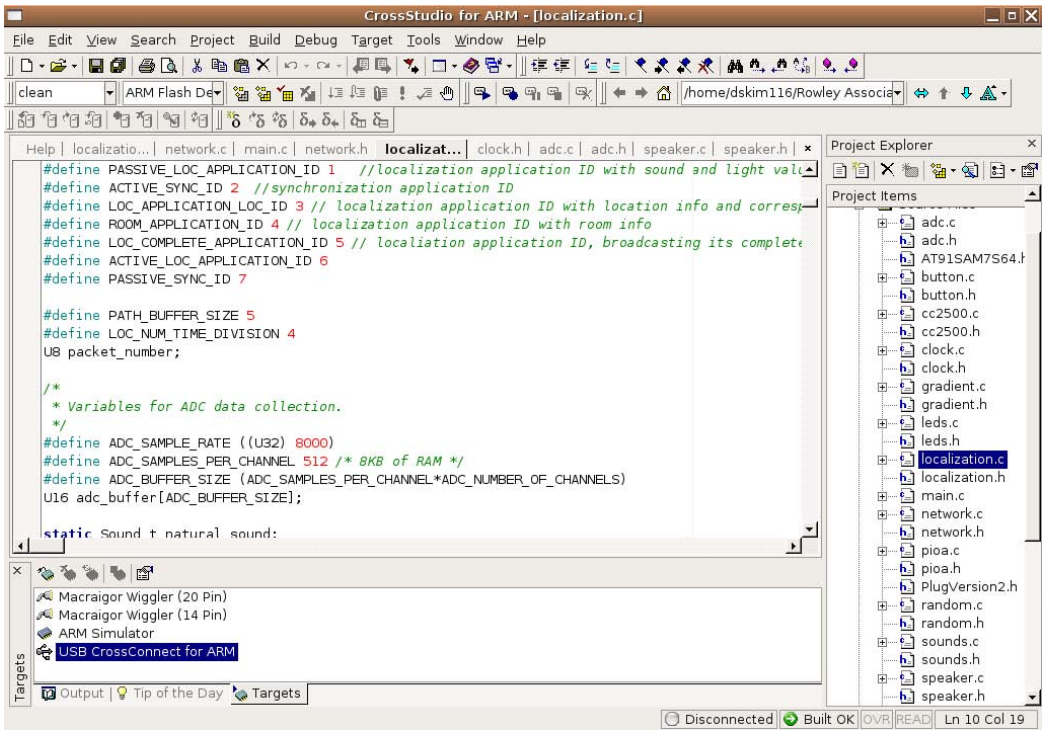


Figure 4-5: Screenshot showing CrossStudio for ARM IDE

you download your programs directly into Flash and debug them seamlessly from within the IDE.

Chapter 5

Localization Based on Natural Phenomena

Most sensory data has only limited utility without location information, and manual node localization becomes impossible for large, inaccessible, or mobile sensor deployments. Accordingly, autonomous localization is crucial for many sensor network applications. Our goal is to develop a distributed localization algorithm for the PLUG indoor sensor network by analyzing sound and light sensory data from naturally occurring phenomena and synthesized emulations of background transient. Our approach has two main phases: passive and active. The system enters an active mode when its sensed region stays relatively silent and stable, hence assumed to be unoccupied. Otherwise, it stays in the passive mode. In the passive mode, each node waits for sonic transients, collects sound sensory data, compares its highest sound peak to synchronized sound peaks from other nodes in its neighborhood and estimates its distance. In the active mode, each node occasionally generates recorded mimics of natural phenomena, such as sonic transients (e.g. pencil dropping or water glasses clinking), or manipulates an attached light source.

As explained in Chapter 4, the PLUGs are subject to conservative resource limitations, with limited processing power and memory. Thus, an attempt has been made to avoid intensive computation and memory use, but to still obtain decent accuracy in localization. Here, the author would like to make a distinction between a rang-

ing scheme and a localization system. Our ranging scheme, which consists of active and passive schemes, estimates pair-wise distances, whereas the localization system localizes each node based on the set of pair-wise distances. The active localization refers to the localization system employing the active ranging scheme, whereas passive localization refers to the localization system employing the passive ranging scheme. Although the ranging schemes are executed on the PLUG, the localization algorithm runs on the base station; this explains why the author's algorithm lies directly in the **third** network category, which was discussed in Chapter 2.

5.1 Objective

The localization algorithm needs to estimate each PLUG's location based on pair-wise distances and display it on a building map. Decent accuracy needs to be achieved over reasonable time periods up to a simple reflection for both active and passive mode. Then, it should be able to display this information on a map on the base station's host PC. In addition, PLUGs need to determine whether they are in the same room or not based on the light sensory data and perhaps infer they are on the same surface or not from the vibration sensor data, which also needs to be displayed on the base station's host PC.

5.2 Problem Statement

Distances are estimated using a TDoA scheme based on either recorded mimics of natural sound or random sonic transients. Although the author does not have any control over the random sonic transients, the author does have control over recorded mimics of natural sound. These mimics need to be loud enough to have decent detection range but soft enough to be relatively unobtrusive. As pointed out earlier in Chapter 2, fine synchronization is important for the TDoA scheme. Because the pair-wise distance estimation relies on synchronization, synchronization is needed between all nodes that can detect the sounds, e.g. all nodes in the neighborhood.

The localization algorithm needs to produce correct relative location with correct rotation and translation. The localization scheme is targeted for both room-scale use and floor-scale use, and these two different targets need to be implemented differently. Although the experiment is done only for room-scale localization, the idea of how to expand it to floor-scale localization is discussed in Chapter 7. In this chapter, only the room-scale localization is discussed. In addition, the application needs to estimate the approximate location of the room with estimated size based on the collected light sensory data and the estimated location of PLUG.

5.3 Assumptions

As pointed out earlier, there are two reference PLUGs which already know their location before the algorithm executes. One of these PLUGs is connected with hardware to the PC for data logging purposes. Note that this PC will be referred to as the base station throughout the thesis. It is assumed that a PLUG does not have more than 16 neighbor PLUGs, which is a reasonable assumption. As a comparison, Broxton [8] made the same assumption about the number of neighbors in an even denser sensor network test bed, 60 nodes over $1m$ by $1m$ area, and still achieved a reasonable localization accuracy. Different PLUGs are assumed to be in the same room if they detect a similar light change pattern almost simultaneously within radio reception range. Although this is not always necessarily true, the author believes that these interpretations are reasonable to deduce room boundaries from the light sensory data. Unlike research projects introduced in Chapter 2, moving object location is not estimated.

5.4 Procedure

Before the localization algorithm executes, each PLUG transmits its UID to its neighbors, and this is performed on the network layer. Depending on the environmental states, each PLUG enters either active mode or passive mode. The PLUG enters active mode when it senses "silence" for certain time, meaning that the room is probably

unoccupied, much like a cricket that chirps when animals are not nearby. Of course, in a perfect world, sensory data should not change at all in "silent environments." However, this is not true in the real world, where there is always some level of background noise. Hence, the possible noise in sensor measurements needs to be taken into account. Thus, the environment is defined to be "silent" if each sampled sensory data has not changed over $\pm$ 10% from sampled sensory data at the previous time slot for approximately 30 minute continuously (a discriminant can also be applied to average background noise). Otherwise, the PLUG enters the passive mode. When the entire network determines that the environment is "silent," PLUGs perform active localization.

Every PLUG keeps track of the set of its neighbors and the estimated corresponding pair-wise distance to them. Based on the results from active and passive ranging schemes, each PLUG updates the set of pair-wise distances by a simple averaging operation. More detail on this algorithm is discussed in Section 5.8. Upon request from the reference PLUG, which is triggered by the base station, every PLUG transmits its set of pair-wise distances to the base station over a pre-determined network gradient. This network gradient is also created in the network layer before the localization algorithm begins. More detail on how the gradient is managed is presented in Chapter 4. In addition to the pair-wise distances, every PLUG keeps track of its estimated room index and transmits its room index information to the base station in a similar fashion over the pre-determined gradient, where the room index is estimated from light sensor data. As the base station receives the pair-wise distances and room information from PLUGs, and if it has enough distance information to localize the PLUG, it displays the location of the PLUG with corresponding room index on the map.

5.5 Preliminary Methods

The method described in this section was ruled out after careful consideration and implementation. However, the author believes that it is still worth mentioning. For

easy explanation in this particular section, the author arbitrarily picks any two non-reference PLUGs and name those PLUG A and B. PLUG A collects sound samples for 0.5 seconds and divides the sound wave into 4 time frames of equal length. Then, the peak sound value is picked from each time frame with the corresponding timing. PLUG A transmits 4 peak sound values with corresponding timing to its neighbor PLUG B. PLUG B estimates the pair-wise distance between A and B by comparing its own 4 peak sound values and corresponding timing with those of PLUG A. Transmitting the entire waveform would be ideal to calculate the delay. But this would result in excessive network traffic and delay in the microcontroller, which certainly are not wanted. Hence, the author chose 4 time frames to avoid network traffic, but to calculate sound delay accurately. However, this did not perform well. As shown in Figure 5-1, the author took an average over many samples. Although the average sound delay seems to converge to the right values, it still contains many errors after many trials. Based on these results and those of his colleagues [8], the author devised different acoustic approaches to estimate the distance. More details on each algorithm are found in the following sections.

5.6 Ranging scheme: Active Ranging and Passive Ranging

Both active localization and passive localization have been examined in depth in many previous studies. However, to the author's best knowledge, no work has focused on the combination of these two with practical assumptions. Many approaches to active localization have been studied under the assumption that every activity is controlled. Global pinging on the entire network is one popular example of controlled environmental activities, which usually localizes rather easily using triangulization techniques. However, this assumption is rather impractical for extended indoor environments, where numerous random events happen. The author believes that it is possible to control a small subset of the network environment instead of the entire network and

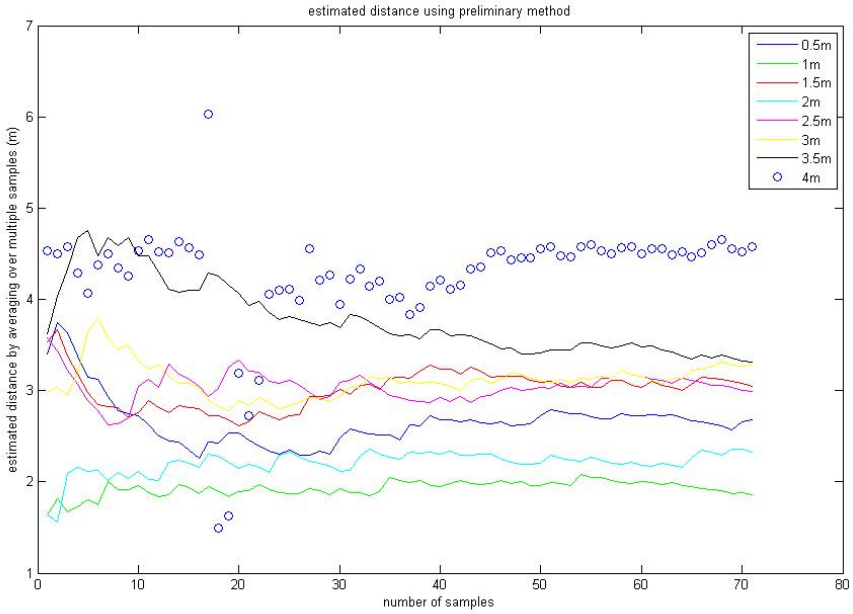


Figure 5-1: The estimated pair-wise distance average over many samples using preliminary methods which were ruled out for its poor performance. x-axis represents number of samples taken and y-axis represents the estimated distance in m.

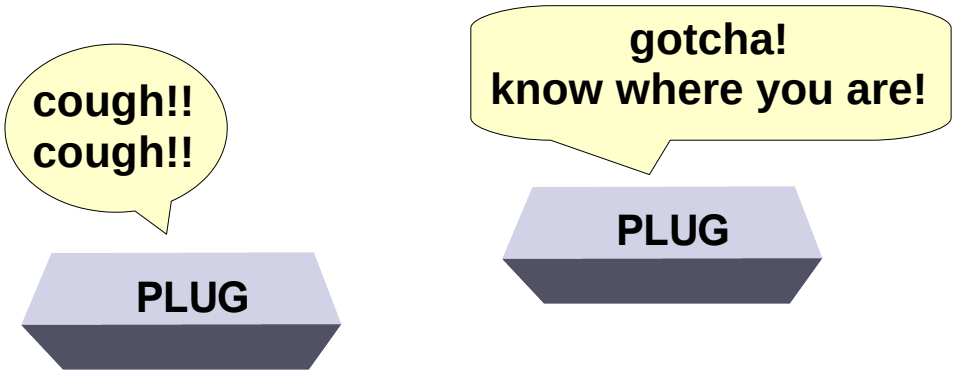


Figure 5-2: Active Localization: When no activity has been recorded for certain time, the PLUG network goes into active-localization mode. In this mode, during every allocated time slot, one PLUG makes one of several recorded natural sounds, such as coughing, and other PLUGs listen. Every action is done in synchronized manner.

decided to base his ranging algorithm on local acoustic activity. By contrast, for passive localizations, although the environmental activities are not controlled, the system assumes that there is only one unique event in its neighborhood. Overall, active ranging performs better than passive ranging because every "controlled" event does not have as much randomness as a "natural" unique event does. But it is not realistic to always assume that environment is always controllable enough to use active ranging. The author proposes to use these two schemes opportunistically and claims that this combination is applicable to the indoor setting for localization. More precisely, as shown in Figure 5-3, the active ranging executes when the environment is assumed to be unoccupied and silent; otherwise, the passive ranging executes. Figure 5-3 also presents when the room index for each PLUG is determined. More detail on room index determination is shown in Section 5.6.2. Their performance is shown in Chapter 6.

5.6.1 Active Ranging

Because the environment is found to be silent, hence controllable, it is easy to devise active schemes. The author proposes three ways to control the environment.

Mimics of Natural Sounds

Each PLUG is initially preloaded with two what we term "natural sounds": a pen dropping sound, called "tung" (index 0) and water glass clinking sound, called "chalang" (index 1). Both of these sounds were recorded on a PLUG in 10 bits with a 4KHz sampling frequency. Each PLUG takes a turn to make the designated sound, from PLUGs with smaller UID to those with larger UID's. During each allocated time slot, one designated PLUG 1) randomly chooses the index for the natural sound to play, 2) broadcasts a packet with its UID and the index of the chosen sound for synchronization purposes, and 3) makes a sound. During that allocated time slot, other designated PLUGs enter listen-mode. In listen-mode, a PLUG 1) receives an index of the chosen sound while it waits for two seconds to hear from its designated

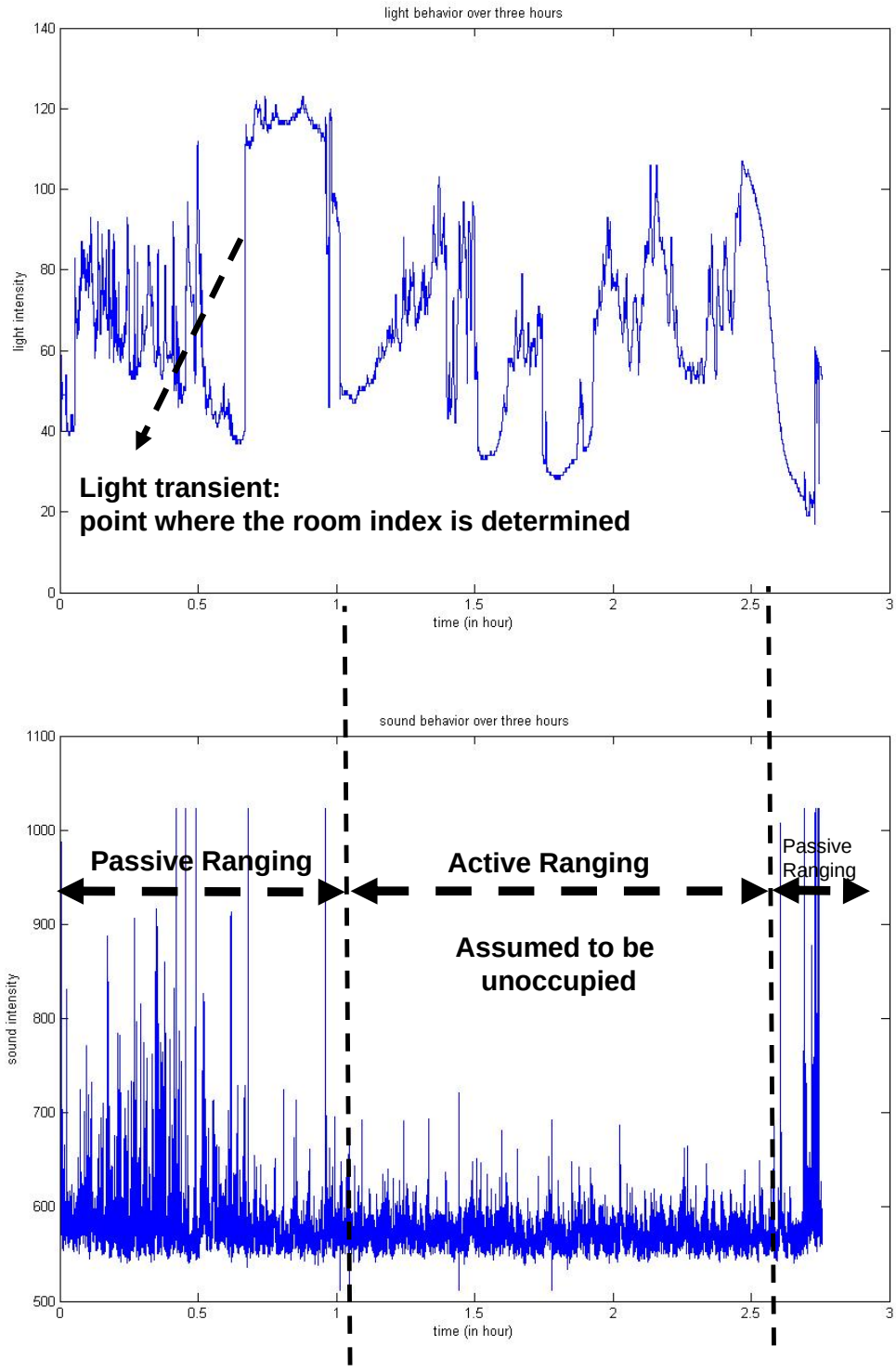
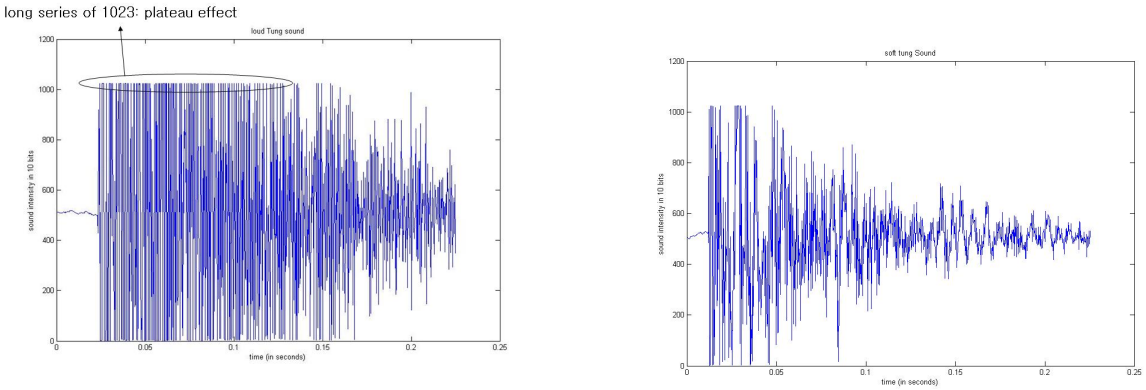


Figure 5-3: Light and Sound behavior over three hours: As explained in Section 5.6.2, the room index is determined when the light transient happens. ranging scheme alternates between active and passive depending on whether the surrounding is silent or not.

PLUG, 2) collects the sound sensor data and runs a matched filter against the corresponding preloaded sound, 3) calculates the time delay from the matched filter's cross correlation peak, and 4) calculates the pair-wise distance to the sound origin. These PLUGs in listen-mode calculate the time delay by a matched filter scheme, which is discussed in Section 5.6.1.



(a) Loud "tung" sound, clipped sound (b) Soft "tung" sound, unclipped sound

Figure 5-4: Due to its *plateau* effect, part (a) was initially ruled out and part (b) was used. However, to increase the detection range, the loud "tung" sound, part (a), was decided to be used.

Initially, the author used the soft "tung" sound to estimate the distance. The soft "tung" sound was intentionally recorded to avoid the saturated *plateau* at the top of the sound wave. If the sound is too loud, a long series of 1023, the highest sound sample value is presented, hence a *plateau* effect happens, as indicated in Figure 5-4. The *plateau* effect makes the "tung" sample sound relatively more artificial than the one without the *plateau*. However, in the soft "tung" sound case, the detection range was only about 20 cm, which is too small for the indoor localization environment. However, the loud "tung" sound gives a detection range of over 2.5 m. Accordingly, the author decided to use the loud "tung" sound. Following a similar process, the "chalang" sound, as shown in Figure 5-5, was picked. With a better amplification and speaker system, or a more selective audio pre-filter and omni-directional microphone, the sound without saturated *plateau* would probably work at much larger range.

Note that, although we term them "natural" sounds, dropping pencils and glasses

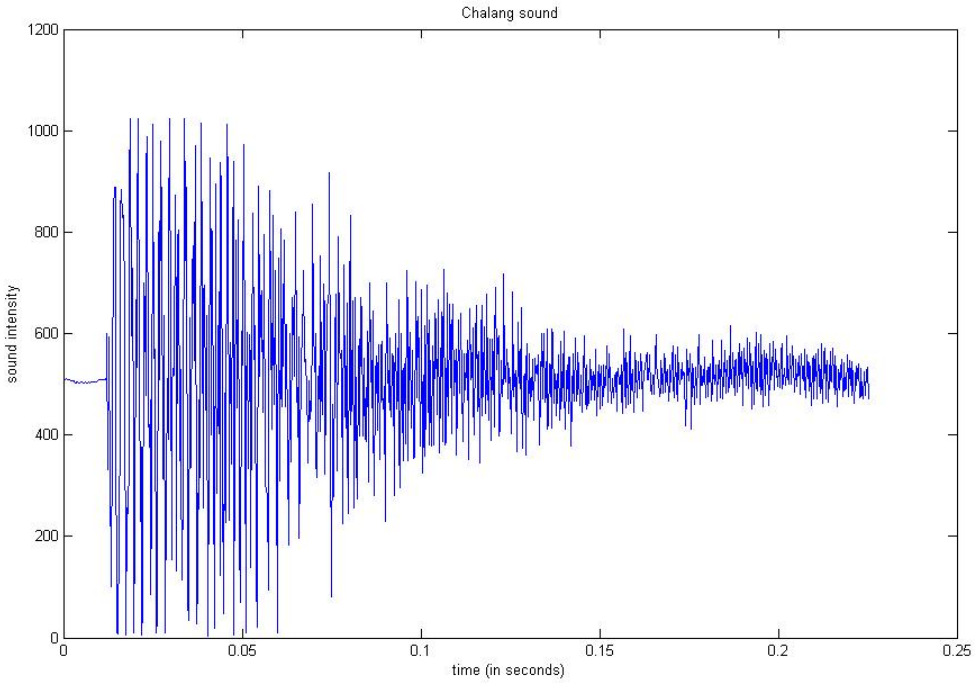


Figure 5-5: "chalang" sound when two glasses touch

clinking may not be natural for most environments. Our algorithm prefers sounds with a fast attack transient. Many other sounds fit this criterion; with more onboard memory, a wider variety of sound samples can be stored, including sounds that may be more natural and subtle in different deployed environments.

Estimating time delay using a Matched Filter

The estimated distance should simply follow Equation 5.1:

$$\text{distance} = \text{speed\_of\_sound} \times \text{estimated\_time\_delay} \tag{5.1}$$

The time delay can then be calculated using the matched filter (see Figure 5-6 for illustration). The PLUGs calculate the cross-correlation of the delay-shifted version of the recorded sound wave and received sound wave for every possible delay value. They look for the maximum cross-correlation value and corresponding delay value in the quadratic interpolated graphs, which is the best-fit quadratic graph across

the cross-correlation values with minimum least square error. Three cross-correlation values, y_1, y_2, and y_3 with corresponding time stamps, x_1, x_2, and x_3, are needed to devise the interpolation and after many trials, the author finds the following way to be the optimum. y_2 is the highest peak cross-correlation value with x_2 being the corresponding time delay. $x_1 = x_2 - 1$, one unit time earlier than x_2, and y_1 is the cross-correlation value corresponding to time delay value x_1. As x_3 is larger than x_2, (x_3, y_3) is the point right before the cross-correlation values start increasing. An alternate way, more mathematically robust but not necessarily showing better performance, is presented in Appendix B. After careful calculation, the coefficients for the quadratic interpolated graph, $y = ax^2 + bx + c$, turn out to be following:

$$\begin{aligned} a &= \frac{(y_1-y_2)(x_2-x_3)-(y_2-y_3)(x_1-x_2)}{(x_2-x_3)(x_1^2-x_2^2)-(x_1-x_2)(x_2^2-x_3^2)} \\ b &= \frac{(x_2^2-x_3^2)(y_1-y_2)-(x_1^2-x_2^2)(y_2-y_3)}{(x_2^2-x_3^2)(x_1-x_2)-(x_1^2-x_2^2)(x_2-x_3)} \\ c &= y_1 - ax_1^2 - bx_1 \end{aligned}$$

Although the speed of sound changes according to the temperature of the medium, the author decided to use 34800 cm/s for the speed of sound, which corresponds to the speed of sound at 28 °C. The correlation value is maximum at $-\frac{b}{2a}$, which is the estimated delay time. The estimated distance is $-\frac{b}{2a} \times \text{speed\_of\_ sound} \times \frac{1}{\text{sample\_frequency}} = -\frac{34800b}{4000 \times 2a}$.

Artificial Natural Light

The PLUG is able to switch on/off a light source attached to any of its outlets. Based on the measured current pattern, the PLUG can usually guess whether the attached device is a light source or not. All PLUGs with a light source attached take turns, switching on/off a light, from the PLUG with the smaller UID to that with a larger one, in similar fashion that artificial natural sounds were created. During each allocated time slot, one designated PLUG 1) broadcasts a packet with its UID, 2) waits for two seconds and then 3) switches on/off a light. During that allocated time slot, other designated PLUGs are in listen-mode. In listen-mode, a PLUG 1)

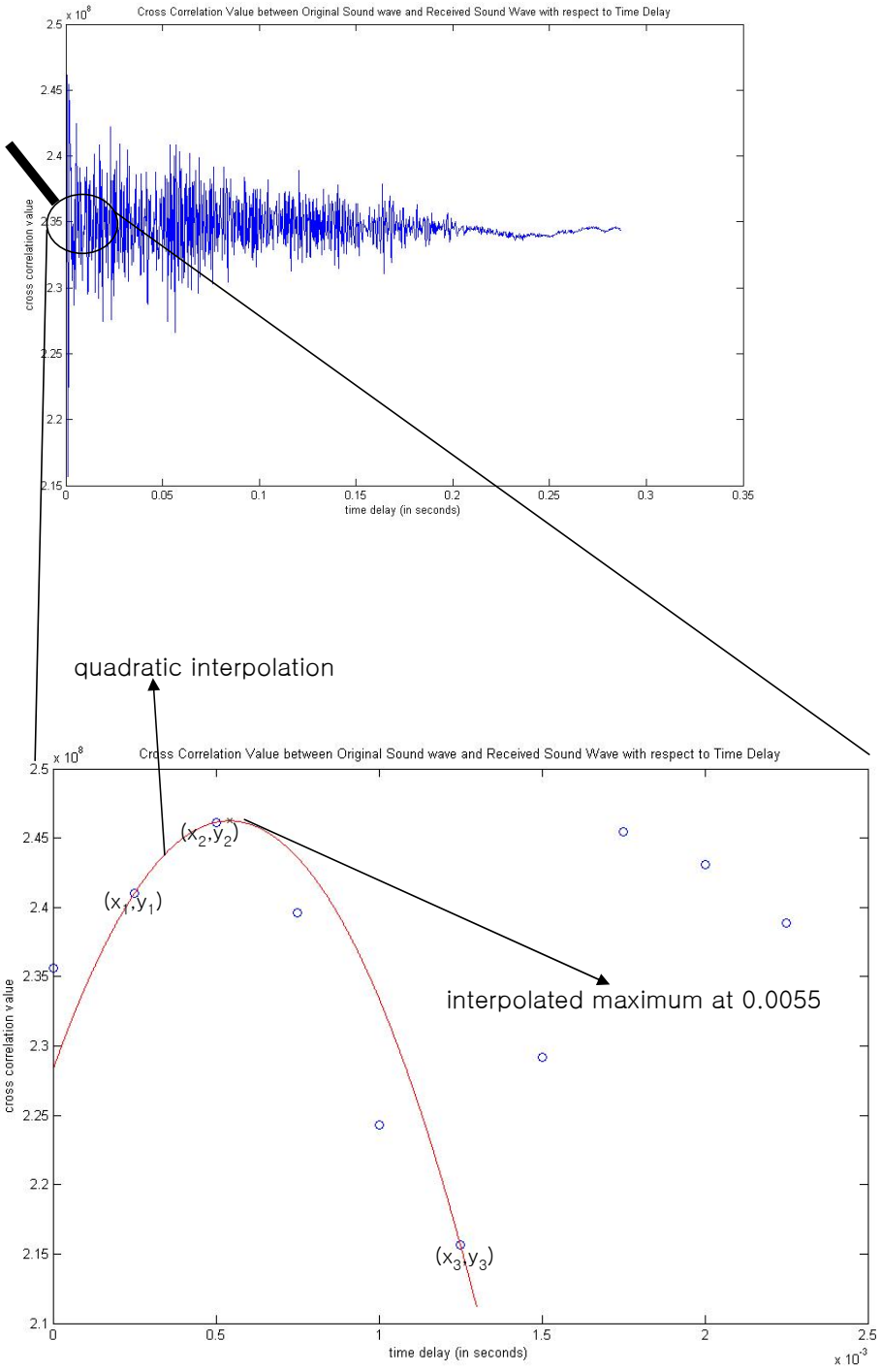


Figure 5-6: Cross correlation value when speaker and microphone are 20 cm apart. Substituting values into Equation 5.1, the estimated time delay should be $\frac{20cm}{34800cm} = 0.00057$. As shown, the expected time delay is close to the estimated one with $0.00002s = 20\mu s$ difference.

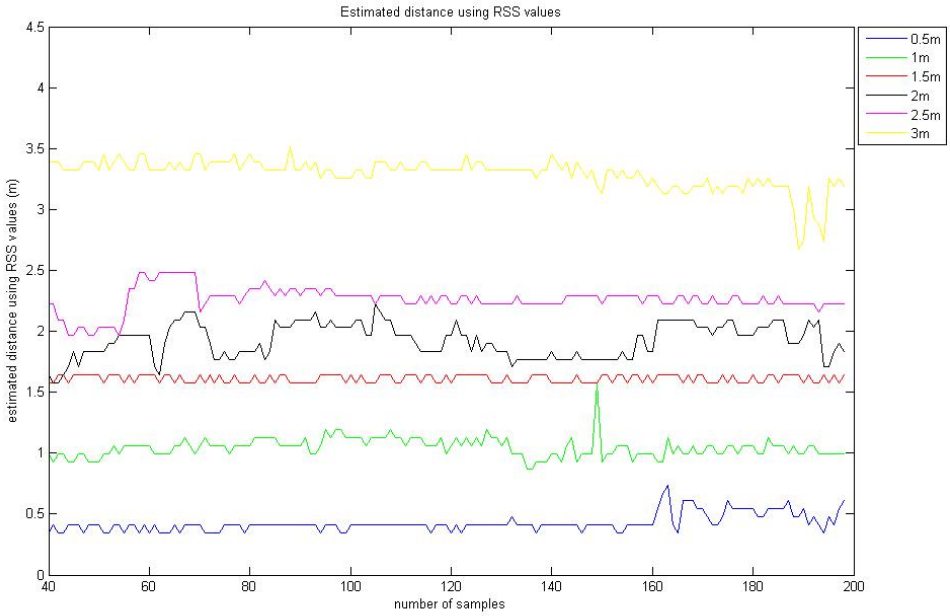


Figure 5-7: Estimated distance using Equation 5.2 and RSS values for different pair-wise distance values in relative radio silence.

receives packets while it waits to detect its designated PLUG turning on/off light , 2) expects to detect the light behavior after two seconds, and 3)concludes whether both PLUGs are in the same room or open area. However, the implementation had poor performance, partially because determining whether a light source is attached or not to a PLUG can be erroneous. Instead, the author decided to use the light sensory data for room estimation in passive mode, where all nodes compared results with one another when they encountered a common change in ambient light.

Radio Signal Strength (RSS)

This is one of the popular techniques used in ranging schemes for sensor networks and has been extensively studied in RADAR [3] and Ecolocation [100], both of which were presented in Chapter 2. First, let us look at the bright side of RSS (Radio Signal Strength), namely its ranging technique. As shown in the graph in Figure 5-7, the RSS can sometimes be a good indicator of pair-wise distance in relative radio silence and in the absence of significant multipath effect: log RSS values are seen to decrease

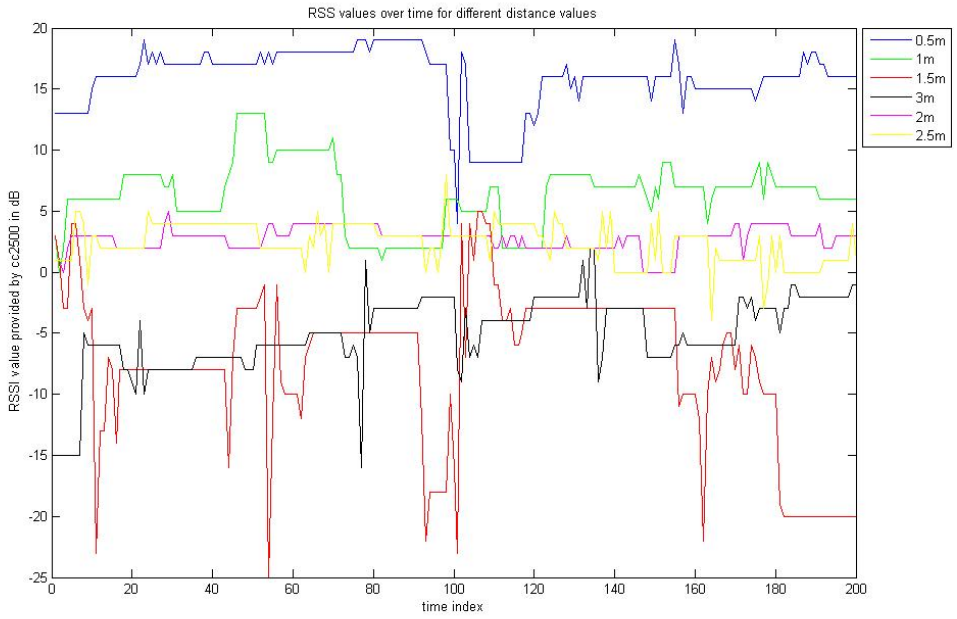


Figure 5-8: RSS values for different pair-wise distance values when a cellular phone is nearby. The noisy RSSI values most probably result from other radio signals and multipath effects.

almost proportionally as the pair-wise distance increases. Let us review the equation on RSS (in dBm) vs distance presented in [100]:

$$RSS(d) = P_T - PL(d_0) - 10\eta \log_{10} \frac{d}{d_0} + X_\sigma - C \tag{5.2}$$

where, P_T is the transmit power and $PL(d_0)$ is path loss for the reference pair-wise distance of d_0. η is the path loss exponent and the random variation in RSS is expressed as a Gaussian random variable of zero mean and σ^2 variance. C is a constant that depends on the type and number of obstructions between the transmitter and the receiver. The values for each variable are summarized in Table 5.1.

The results from Artificial Natural Light method can roughly determine whether C is zero or non-zero, which can be helpful estimating the value of C. However, estimation of C is not enough. Other parameters need to be accurately measured, and this requires careful calibration beforehand. Even with careful calibration, the localization has rather poor accuracy (approximately 3m) in an indoor environment.

Parameter	Typical value
P_T	4dBm
$PL(d_0)$	55dB
η	4 (indoors)
σ	7 (indoors)
α	25
λ	15
$\beta(=\frac{\alpha}{\lambda^2})$	0.11
γ	0.1
ρ	8

Table 5.1: Values used for RSS equation parameters

RSSI is susceptible to external biases such as interference, shadowing and multipath effects, as well as environmental variations such as changes in temperature and humidity [72]. These physical effects are difficult to predict and depend greatly on the actual environment in which the system is operated. Because RSS values are sensitive to noise, as shown in Figure 5-8, and multipath, they are not robust to estimate the pair-wise distance, thus the author decided to rule out this RSS option.

Final Remark

Among the three active ranging schemes mentioned, the author decided to use only sound. The comparison between active ranging using sound and that using RSS values is presented in Chapter 6. Table 5.2 summarizes the active ranging scheme. Here, the "update" process is a simple average based on many estimates. The algorithm keeps track of the number of estimates, current distance estimation, and average. When it takes new estimation, it calculates the new average and replace these variables accordingly.

5.6.2 Passive Ranging

In passive mode, because the environment is not controllable, each PLUG does not know which sound it will hear or which light pattern it will detect. The author believes that the best we can do in passive ranging with optimum network traffic loading is to look for the sharp transients in the collected sensor data. For sound, each PLUG

ACTIVE_RANGING
for `i=1` to MAXUID if CURRENT_UID == `i` goto ACTIVE_PLAY else goto ACTIVE_LISTEN
ACTIVE_PLAY
1. Choose sound index `k` randomly among $\{1, 2\}$. 2. Broadcast a packet with UID `i` and `k`. 3. Make Sound of index `k`
ACTIVE_LISTEN
1. Receives `UID i`, and `k`. 2. Records the artificial sound. 3. For each possible time delay, calculates the cross-correlation between the recorded sound and preloaded sound of index `k`. 4. Looks for the largest cross-correlation and its corresponding time delay. 5. Estimates the pair-wise distance using time delay and updates the pair-wise distance.

Table 5.2: Active Localization Algorithm

transmits the timing of the first significant sound peak in the sonic transient. Note that the sample is not divided into 4 time frames to choose the 4 sound peaks from each time frame as in Section 5.5. Only one sound peak is chosen from entire period for ranging and it is found to perform better than that preliminary method. Based on the change in light sensory data, each PLUG estimates the room index.

Sound

In passive mode, as soon as any PLUG detects a sonic transient, it synchronizes the neighborhood and declares itself as a temporary leader. On receiving the synchronization packet, every other PLUG samples a 0.4 second interval of sound, then also looks for the peak in their sample and stores its corresponding timing. The relative timing is assumed to be a differential time delay, and based on this delay, the PLUG estimates the pair-wise distance to the temporary leader simply based on the Equation 5.1. Here, it is assumed that there is no saturated *plateau* effect in the sampled sound: under the saturated *plateau* effect, the time stamp might not necessarily

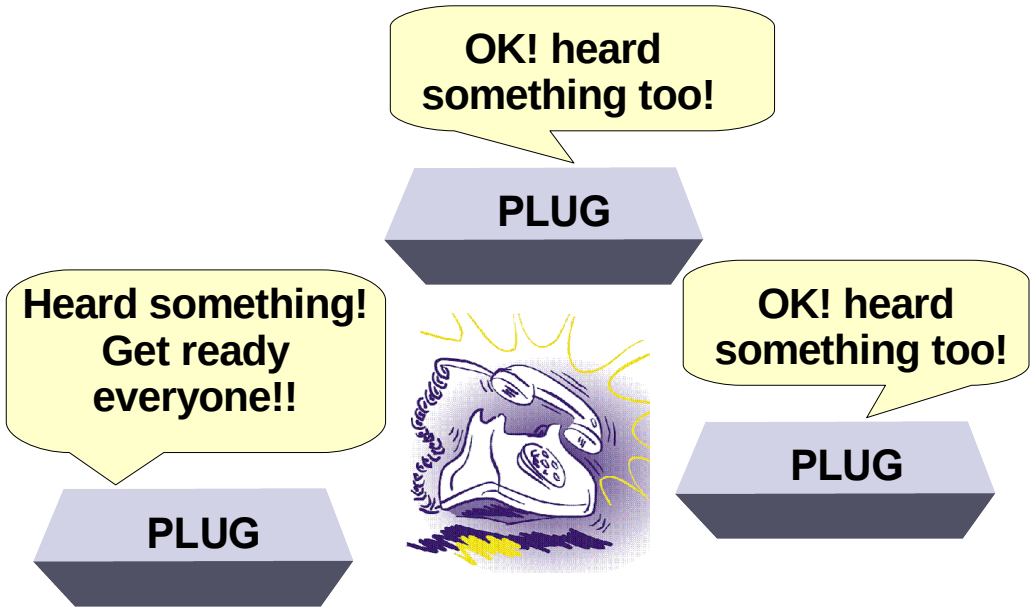


Figure 5-9: Passive Localization

represent the proper time delay.

Now, let us look more carefully at how the time delay value is estimated. For easy explanation, let us call the temporary leader PLUG "A" and one of its neighbors "B." The time delay is maximum when the sound source, PLUG A and PLUG B are in line; the time delay is minimum when the sound source is equally distant from the PLUG A as from the PLUG B. Thus, the maximum value among the collected time delay values should be equal to the distance between the PLUG A and B, scaled by the speed of sound. However, the maximum value is sensitive to noise and multi-path, hence caution needs to be taken into consideration. Figure 5-10 shows the distribution of the collected sound delay values; the data were collected from busy areas in one of MIT college dorm for 20 hours. As shown in the graph corresponding to the values between node 4 and 6, simply picking the maximum might distort the estimation. In this case, the maximum is some value between 35 and 40, but it would be ideal to pick some value between 20 and 25 instead of the value between 35 and 40 because the value between 35 and 40 is at an extreme tail and probably results from multi-path. The author decided to "intelligently" pick the maximum time delay by choosing it only

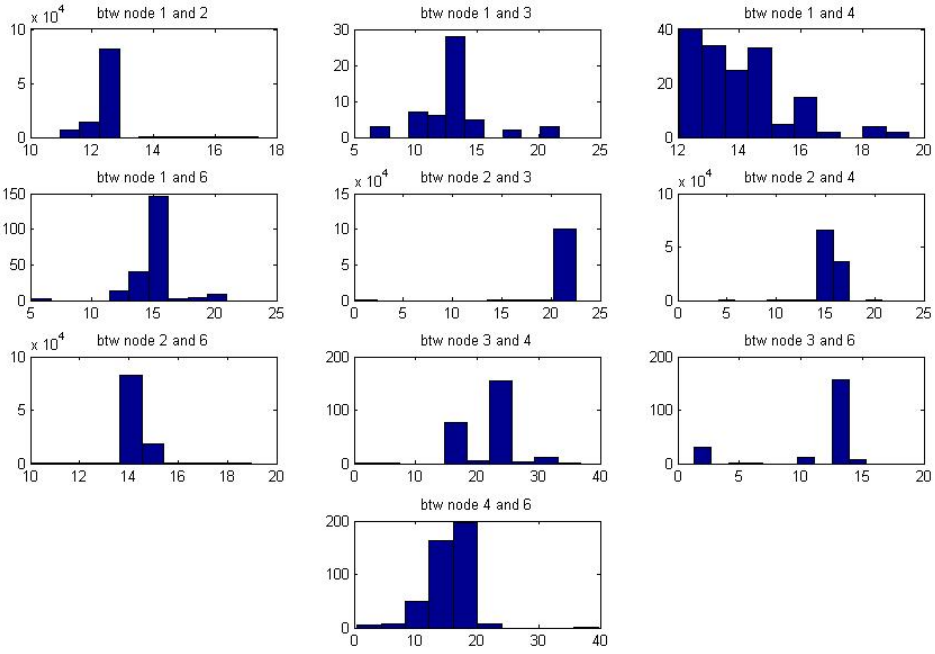


Figure 5-10: Distribution of the collected delay values. x axis values represent time delay values, where unit time delay corresponds approximately to $9cm$.

if the maximum value is within three standard deviations from its mean (after many trials, three standard deviations perform better than other choices). This method is empirically proven to perform well; the performance comparison between different methods are shown in Section 6.5.2.

It would be ideal if each PLUG could transmit/receive the entire sampled sound wave. Then, each PLUG would be able to estimate the distance using the matched filter as in the active ranging scheme. However, this would result in massive network traffic unless the entire sound wave can be compressed into few constants, e.g. wavelet coefficients. These few wavelet coefficients might be sufficient to estimate distance more accurately. In fact, there have been several studies in medical and sonar fields, detecting and classifying the received sound through a transient model using few wavelet coefficients [88] and this can be an interesting extension of the author's thesis work.

In addition to the delay value, the amplitude of sound peak could be a reasonable

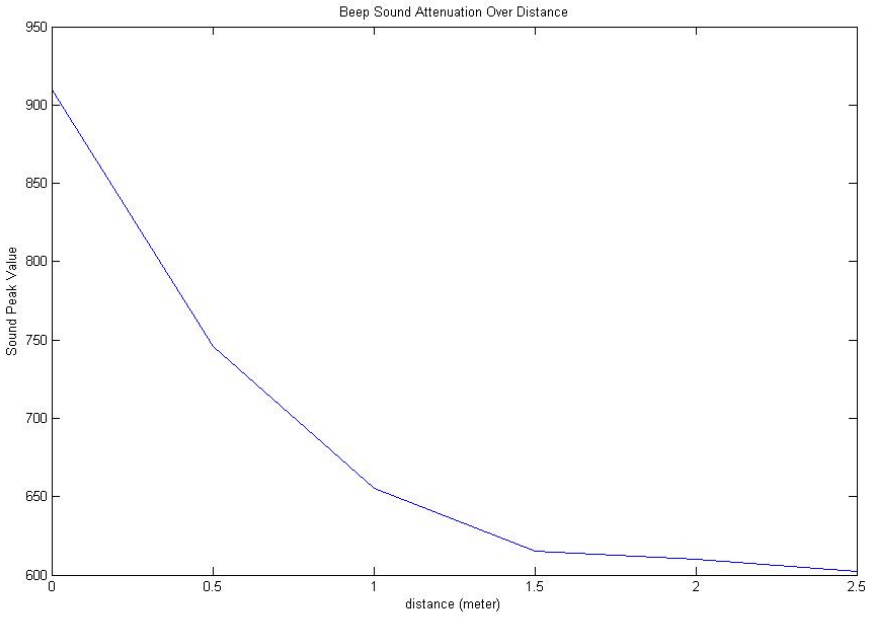


Figure 5-11: Sound Peak value vs pair-wise distance

measure to help estimate the pair-wise distance as it attenuates with range as shown in Figure 5-11. As expected, the sound peak value decreases proportional to the squared pair-wise distance. Thus, the peak values can be compared between neighboring PLUGs to make the estimation more accurate. Despite its great performance in this controlled situation, the peak values turn out to be sensitive to the noise, outliers, saturated *plateau* effect, and multipath effect. Thus, the author decided not to use peak value for his ranging scheme. For performance comparison, ranging results based on peak values are presented in Chapter 6.

Light

The room indices are initialized to be the PLUG UIDs. Once neighboring PLUGs determine that they detect the similar light patterns in the approximately same time slot, they agree on the same estimated room index, modify their room indices accordingly, and transmit the room index to the base station for updating.

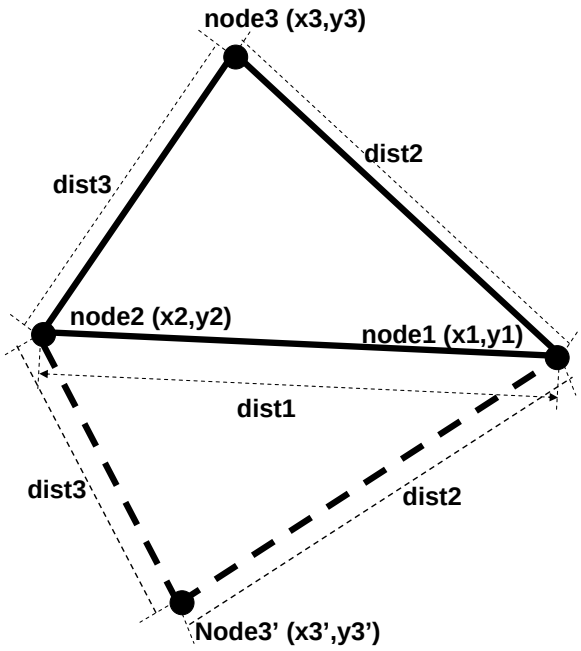


Figure 5-12: For easy illustration of linear localization algorithm

5.7 Localization Algorithm: average-lateration scheme

Based on Active and Passive ranging schemes, pair-wise distances are calculated. Now, the locations of the PLUGs need to be estimated and displayed. The author presents a linear algorithm. This algorithm is a standard localization algorithm, similar to the one presented by Broxton [8] and many other authors. However, because not every pair-wise distance is available at the same time, the PLUG is displayed on the map screen only when all necessary information is collected. Although the ranging software is executed on the PLUG, the localization algorithm runs on the base station.

5.7.1 Linear Localization Algorithm: Lateration

The base station keeps track of `dist_info`, the list of received information on pair-wise distances. Every node's location is estimated using a triangulization method, i.e., the location of a new node (`node3`) is determined based on the location of two reference nodes (`node1`) and (`node2`). See Figure 5-12 for illustration. In addition, `dist_info` needs to contain the distance `dist2` between (`node1`) and (`node3`) and

distance dist3 between (node2) and (node3). The node is defined to be a reference once the updated_count exceeds a threshold. Once the base station receives four pieces of information: location of (node1), location of (node2), dist2, and dist3, the algorithm checks whether dist1, dist2, and dist3 can make a triangle by checking the following inequality:

$$\begin{aligned} \texttt{dist2} + \texttt{dist3} &> \texttt{dist1} \\ \texttt{dist1} + \texttt{dist3} &> \texttt{dist2} \\ \texttt{dist1} + \texttt{dist2} &> \texttt{dist3} \end{aligned}$$

The x coordinate of new node is following:

$$\begin{aligned} \texttt{first\_angle} = &\quad \arctan(\tfrac{y2-y1}{x2-x1}) + \arccos(\tfrac{(\texttt{dist2}^2+\texttt{dist1}^2-\texttt{dist3}^2)}{(2*dist1*dist2)}) \\ \texttt{first\_slope} = &\quad \tan(\texttt{first\_angle}) \\ x = &\quad \tfrac{\texttt{first\_slope}*x1-y1-\texttt{second\_slope}*x2+y2}{\texttt{first\_slope}-\texttt{second\_slope}} \end{aligned}$$

The y coordinate of new node is following:

$$\begin{aligned} \texttt{second\_angle} = &\quad \arctan(\tfrac{y1-y2}{x1-x2}) + \pi - \arccos(\tfrac{\texttt{dist3}^2+\texttt{dist1}^2-\texttt{dist2}^2}{2*dist1*dist3}) \\ \texttt{second\_slope} = &\quad \tan(\texttt{second\_angle}) \\ y = &\quad \tfrac{\texttt{first\_slope}*y2-\texttt{second\_slope}*y1+\texttt{first\_slope}*\texttt{second\_slope}*(x1-x2)}{\texttt{first\_slope}-\texttt{second\_slope}} \end{aligned}$$

However, there are two solutions that satisfy the above constraints due to the reflection issue as seen in Figure 5-12; both (node3) or (node3') satisfy the constraints. The author would like to present a simple method to decide between these two. Both (node3) and (node3') calculate average distance to its localized neighbors respectively. The node which has the smaller average distance to its localized neighbors is picked to be the new node's location. Note that each node calculates the average distance to only its *localized* neighbors and not to its *unlocalized* neighbors. The

intuition behind this method is that the node is likely to be physically closer to its network neighbors. Then, it updates the location of `node3` and the `updated_count` attribute for `node3` increments by 1.

5.8 Localization Software

The localization module consists of two sections. One section is uploaded in every PLUG and is written in ANSI C. This part is written based on the software discussed in Section 4.3. The other section is loaded into the base station, which is attached to the reference PLUG, and is written in Python.

5.8.1 Localization Application

`localization.c and localization.h`: This software consists of two parts: active localization and passive localization. These two parts perform tasks as discussed in Section 5.6.1 and 5.6.2. The PLUG stays in passive mode most of the time, except for when every PLUG in the entire network determines that the surroundings are silent.

5.8.2 Location Simulator Interface from PC side

`plugloc.py`: This defines class `node` with attributes `x` coordinate, `y` coordinate, `ID`, `updated_count`(the number of update), `isRef`, and `room_id`. `isRef` is `True` if the node is really a reference PLUG or the node has been updated over the threshold number. Based on the location of the reference PLUGs and received set of distances, it estimates each PLUG's location. Class `nodenetwork` is also defined with the attributes `room_size`, `rooms` (array of room ID's), network `size`, `nodes` (array of node's), `IDs` (array of PLUG ID's), `packet` (location network packet), and `parser` (the parsed information of network packet). This class has a function to update data, to add newly found nodes to the network, to update room indices, to update pair-wise distances, and to calculate the `x` and `y` coordinates based on the set of pair-wise distances. Class `new_dist` is defined to keep track of received network packets that

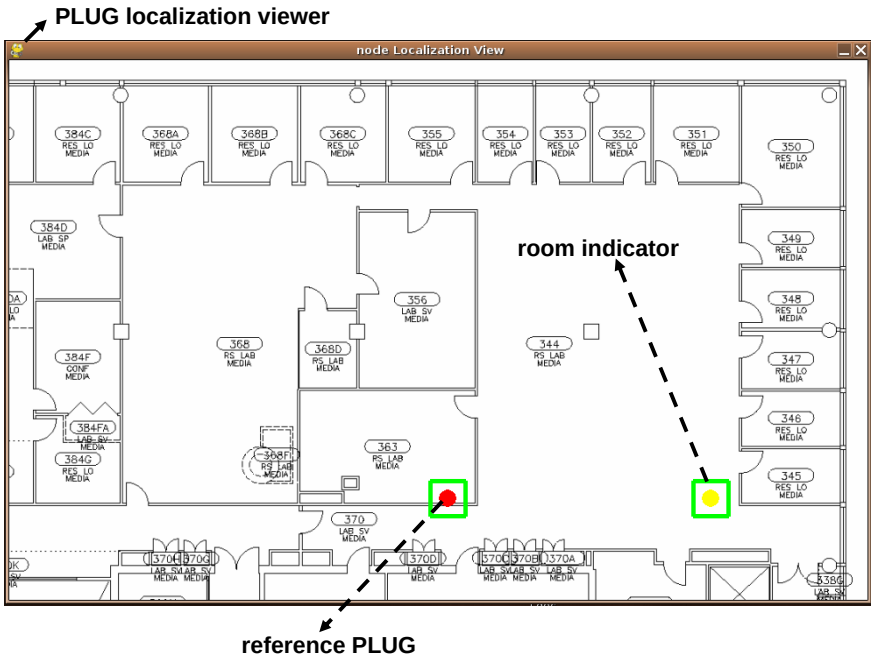


Figure 5-13: PLUG Localization Simulator

have not been processed yet. Class `loc_data_parser` and `loc_ApplicationPacket` are used to parse the received network packets. Global function `visualize` is used display each PLUG's `x` and `y` coordinates in addition to its room ID. If different PLUGs share the same room ID, a square is drawn around them to include these PLUGs. Figure 5-13 is a screen shot from the PLUG Localization Simulator. This simulator has been greatly modified from `PLUGview.py`, which was written originally by Responsinve Environments Group colleagues, Yasuhiro Ono and improved by Josh Lifton to simulate the raw sensory data on the map.

Chapter 6

Empirical Data Analysis

In the following empirical studies, every sound is sampled at 4 KHz. The data for both active and passive ranging are presented. Simulation data for localization using active ranging and that using RSS values are presented and compared. Both passive and active localization data of PLUGs, when they are deployed on the floor, are also presented. It is proven empirically that active localization performs far better than passive localization or localization using only RSS values both in simulation and real deployment.

6.1 Pair-wise distance estimation using Matched Filter: Active Ranging

In active ranging, speakers and microphones are positioned so that they face each other and the pair-wise distance is estimated using a matched filter as discussed in Section 5.6.1. The estimated distance values with corresponding errors are shown in Table 6.1. The average error is $2.1cm$, and this is remarkable performance considering that one sound sample represents $34800cm/s\frac{1s}{4000} \approx 9cm$. The distance estimation performs relatively poorly across short distances. This is attributed to the ambiguous sound source location between two PLUGs due to its bulky size and saturation of the microphone's signal. Furthermore, the matched filter scheme is robust against

Distance (cm)	**Average** (cm)	**Standard Deviation** (cm)
10	10.3208	1.4455
20	26.1864	2.5281
30	28.1696	3.5058
40	37.0939	0.6072
50	46.5141	1.8217
60	56.6780	1.2395
70	66.8419	1.8551
80	83.2032	1.0517
90	87.1696	1.6066
100	102.2915	1.2395
110	112.9511	1.0517
120	121.8755	1.6066
130	130.7999	0.7012
140	141.7075	1.9520
150	151.1276	2.1035
160	161.2915	0.7437
170	170.9596	1.8551
180	180.3797	1.5282
190	186.5772	2.7269
200	199.2201	0.9916
210	208.6403	0.9916

Table 6.1: Estimated distance, averaged over 5 samples, with corresponding standard deviation for active ranging

background noise such as air conditioning, fans, computers, and music. As shown in Table 6.1, the standard deviation is approximately $1.58cm$ on average, which indicates its robustness against background noise.

6.2 Pair-wise distance estimation using Peak Finding: Passive Ranging

In passive ranging, speakers and microphones are positioned so that they face each other, and the sound peak and corresponding time delay are used to estimate distance as discussed in Section 5.6.2. The estimated distance values with corresponding errors are shown in Table 6.2. For experimental purposes, a metal clinking sound was used. The sound source was in line with two PLUGs. The average error per tran-

sient is 50.96cm, which is far worse than active localization's. Nonetheless, this is remarkable performance considering that one sound sample represents $\approx 9cm$ and the environment is not controlled like in active mode. In addition, passive ranging performs a little better than ranging using RSS, which had approximately 57.75cm error on average as discussed in Section 6.3. In passive mode, the estimation performs relatively poorly at long distance. This is attributed to the reason that if two PLUGs are further apart, it is more likely that these PLUGs misinterpret different sounds as being from the same sound source, or the transient peaks differ due to multipath, dispersion, etc. The distance estimation using passive ranging is not robust, as the standard deviation is 158.95cm on average, which is far worse than that of active ranging. The estimation result is more sensitive to background noise and multipath than active ranging. Sound amplitude peak values are also summarized in Table 6.2. As shown in the table, there is no particular correlation between peak values and distance in the real deployment, because the randomly generated sound waves have different sound peaks. Sound peak values can be used in controlled environments where the sound peak attenuates proportionally to the power (2 to 4) of distance. However, passive ranging depends on random sound transients, so does not provide right setting to exploit sound amplitude peaks in distance estimation. The metal clinking used on the sound transient in these tests is perhaps quite ideal for passive ranging based on peak timing; other naturally occurring sound maybe less ideal. As the sound source was in line with the PLUGs, the time delay was maximized and the distance between PLUGs is the speed of sound scaled by the measured time delay. In general, this condition does not hold, hence as explained in Section 5.6.2, a fit was employed to estimate the maximum delay.

Distance (cm)	**Average**(cm)	**Standard Deviation** (cm)	**sound peak**
10	15.78	34.94	1023
20	20.13	60.73	1023
30	33.18	94.38	1023
40	44.06	112.58	896
50	61.46	82.17	896
60	56.02	65.46	891
70	61.46	46.43	839
80	94.09	58.55	897
90	77.78	78.45	890
100	85.39	98.95	725
110	88.65	97.63	1023
120	99.53	93.28	1023
130	115.85	210.55	1023
140	99.53	87.83	1023
150	148.48	105.68	1023
160	159.36	159.29	655
170	235.5	527.15	1010
180	281.18	890.4	773
190	268.13	278.43	633
200	159.36	86.33	853
210	104.97	68.75	1023

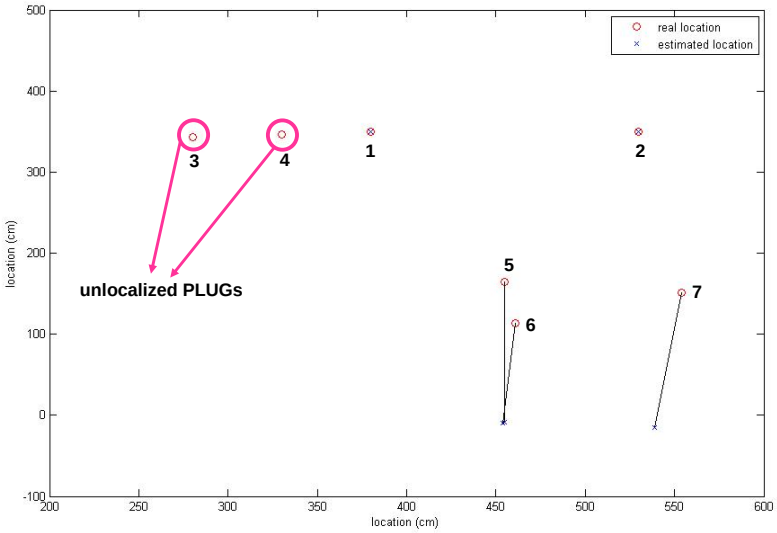
Table 6.2: Estimated distance, averaged over 5 samples, with corresponding standard deviation for passive ranging, in response to metal clinking sound.

Distance (cm)	**Average**(cm)	**Standard Deviation** (cm)
50	2.19	9.65
100	84.92	14.67
150	156.89	7.22
200	365.89	40.55
250	196.94	18.64
300	243.17	15.53

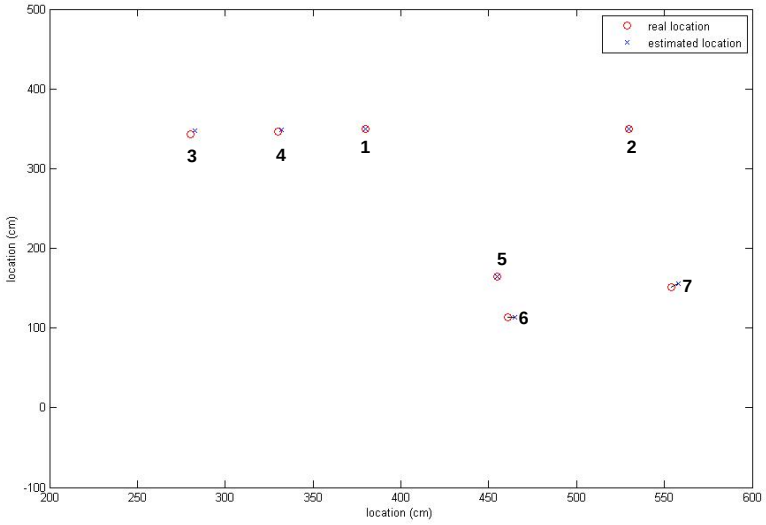
Table 6.3: Estimated distance, averaged over 200 samples, with corresponding standard deviation for distance estimation using RSS.

6.3 Comparison with Simulated Localization Using RSS

The author used a quiet radio environment to collect RSS values to measure distance. Table 6.3 summarizes the pair-wise distance with its standard deviation over The error is 57.75cm on average, which is worse than the pair-wise distance estimation using either active or passive ranging schemes. Here, the author would like to note that this error using RSS can be exponentially worse with only minor radio activity or multi-path effects. As shown in Figure 6-1, localization using active sound has 2.98cm accuracy whereas localization using RSS has 197.68cm accuracy.



(a) Localization using RSS



(b) Localization using sound

Figure 6-1: **(a)** Localization using RSS error analysis plot. This has 197.68cm accuracy. **(b)** Localization using sound error analysis plot. This has 2.9cm accuracy.

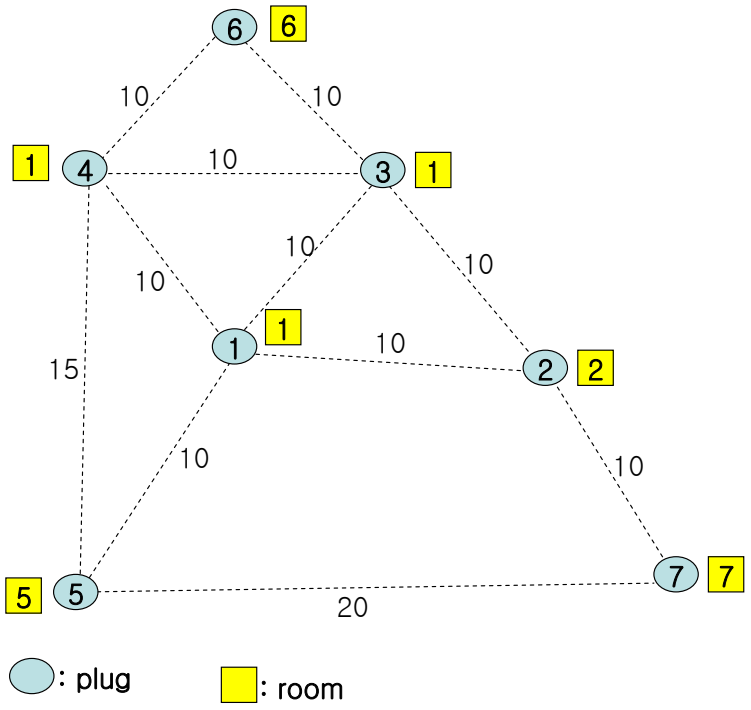


Figure 6-2: Layout for Simulation. The figure is not drawn to perfect scale.

6.4 Simulated Localization Results

To validate the localization algorithm, the author used a valid set of randomly chosen pair-wise distances for simulated localization. As expected, the system performs with no error when there is no measurement error included. The active localization was simulated using the pair-wise distance values from Table 6.1.

6.4.1 Randomly chosen pair-wise distance for localization

Here, pair-wise distances and the number of PLUGs are randomly chosen to simulate a PLUG deployment. As shown in Figure 6-2, the distance between PLUG 1 and PLUG 2 is $10m$. PLUG 1 and PLUG 2 are assumed to be the reference points, meaning that they already know their position before the localization algorithm executes. However, the distance between two reference points still needs to be estimated in order to calculate the factor that is used to scale every pair-wise distance on the map. In other words, the actual $10m$ distance cannot be drawn on the computer

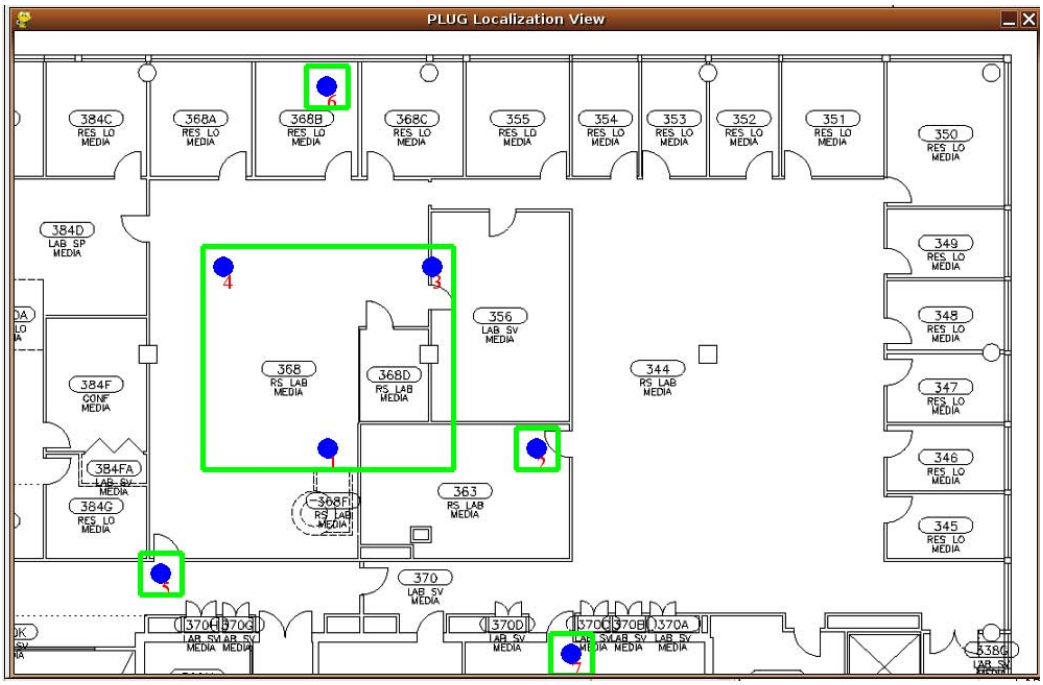


Figure 6-3: PLUG 1, 3, and 4 are determined to be in the same room. PLUG 2, 5, 6, and 7 are determined to be in the different room.

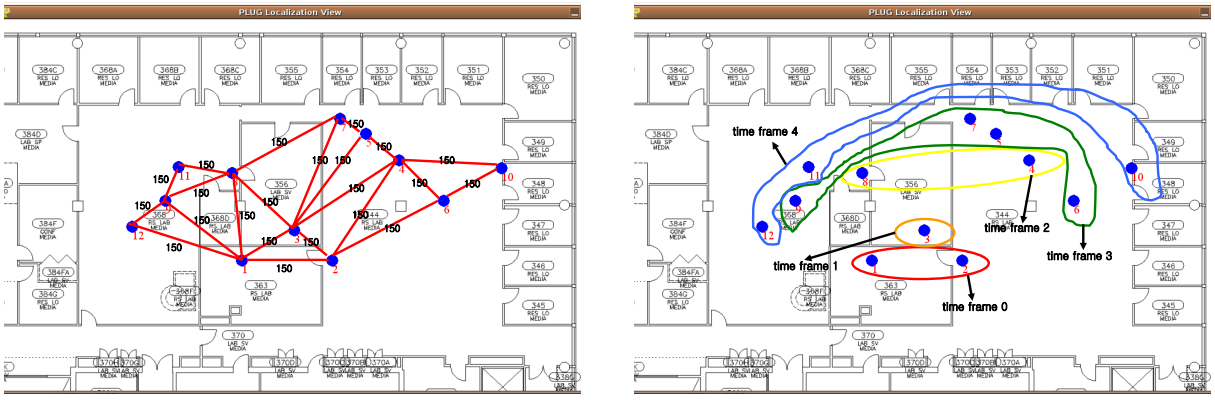
screen; rather, it needs to be scaled down. Based on the initial positions of PLUG 1 and 2, the distance between PLUG 1 and 3, and the distance between PLUG 2 and 3, the location of PLUG 3 is estimated using the lateration scheme as explained in Chapter 5. All the other PLUG locations are calculated using similar methods. The final localization display is shown in Figure 6-3 and its corresponding step-by-step localization is shown in Figure 6-5. As shown in Figure 6-3, the location is estimated with $0cm$ error. This shows that as long as the pair-wise distance estimation is correct, localization is almost error-free except for possible reflection.

Figure 6-3 shows the estimated room shapes based on the estimated location of PLUGs and the detected light pattern. Two PLUGs are determined to be in the same room if they detect a similar light behavior. Once they are determined to be in the same room, a square is drawn around these PLUGs as shown in Figure 6-3.

6.4.2 Simulated active localization

Twelve PLUGs are simulated on $9m \times 6m$ testbed and compared with real locations for error analysis as shown in Figure 6-4. PLUG 1 and 2 are reference PLUGs. Only the estimated distances between direct neighbors are fed to the system for the localization. The set of distances is shown in Figure 6-4 (a). It is worth noting that only the distance between neighbors are available, and this causes significant error propagation, which is discussed shortly.

The average error in the location, using the distance measurements from Table 6.1, turns out to be $7.97cm$. However, as shown in the plot, most of the errors are due to the outliers, PLUGs 8, 9, 11, and 12. Without these outliers, the location error is $5.55cm$ on average, which is comparable to the pair-wise distance measurement error, which is $2.1cm$.



(a) PLUG localization viewer **(b) Localization divided into time frames**

Figure 6-4: **(a)** Simulation of the 12 PLUGs' deployment on $9m \times 6m$ test bed. **(b)** PLUG 1 and 2's locations are estimated initially in time frame 0. Based on these locations, the location of PLUG 3 is estimated in time frame 1. Based on these three locations, PLUG 4 and 8's locations are estimated in time frame 2 and so forth.

Now, let us analyze the error of the outliers more carefully. As shown in Figure 6-4(b), the locations of PLUGs are estimated in a certain time frame. However, locations estimated in later time frames are less accurate than the ones in earlier time frames, and this error is presented separately for each time frame in Table 6.4. This can be explained easily in Figure 6-6: the error propagates over a time frame.

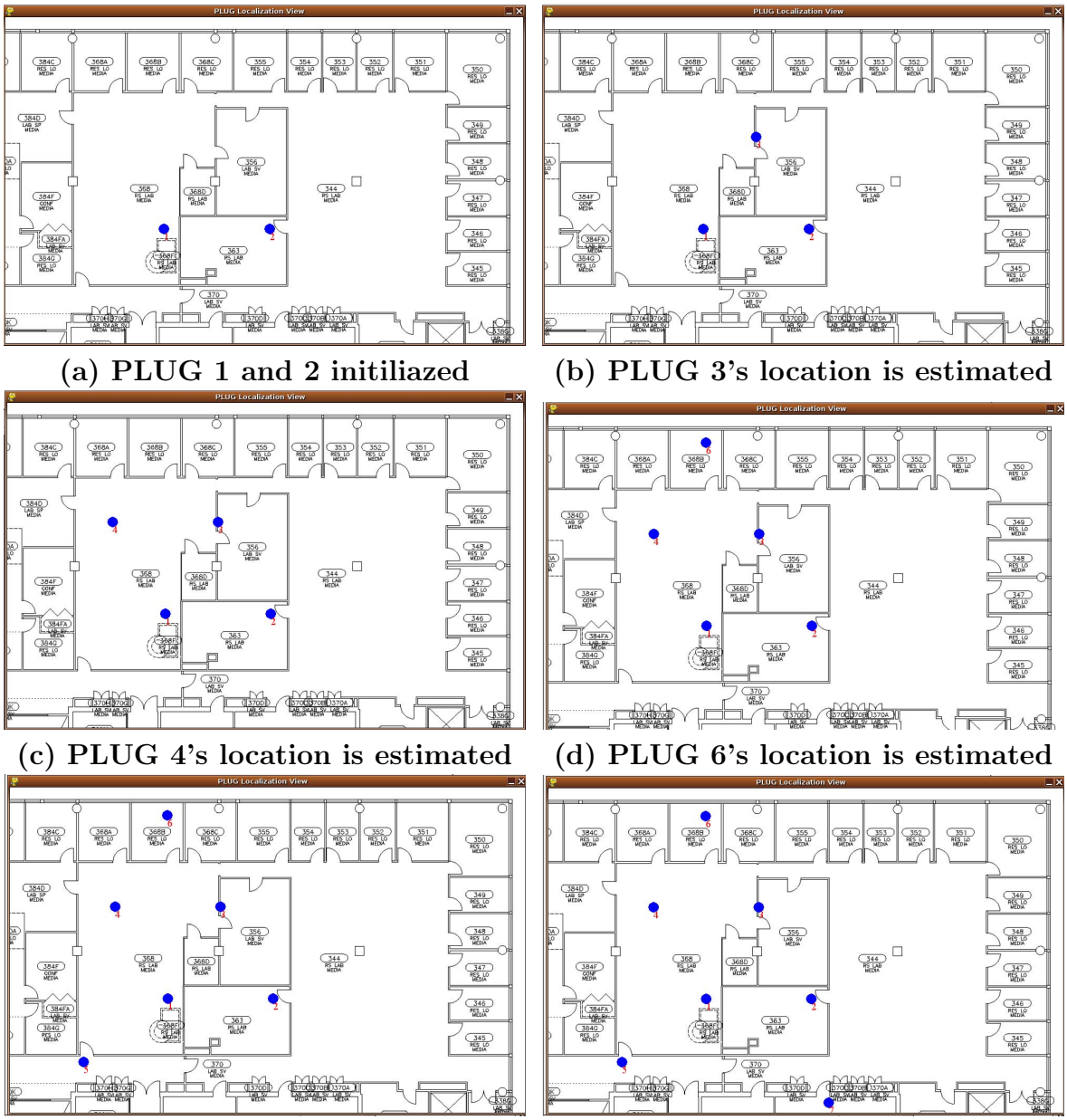
Time Frame	**error in average** (cm)
Time Frame 0	0
Time Frame 1	4.12
Time Frame 2	9.65
Time Frame 3	10.21
Time Frame 4	10.46

Table 6.4: Estimated distance with corresponding errors

Let us consider two PLUGs 4 and 8 in time frame 2. The location of PLUG 4 has a $5cm$ error and that of PLUG 8 has $14.31cm$ error. Because the location error for PLUG 8 is relatively large compared to that for PLUG 4, the PLUGs whose locations are based on PLUG 8 are expected to have worse location estimation than those based on PLUG 4, which is exactly what Figure 6-6 shows. PLUG 11 and 12, which have $10.29cm$ and $15.26cm$ errors respectively, perform worse localization than PLUG 10, which has only $5.83cm$ error.

Because the algorithm is based only on the sound and sound has limited range (about $3 - 4m$), the author's localization is based solely on the local information, which include locations of its one hop neighbors and its estimated distance to one hop neighbors. When the local information is faulty, to author's best knowledge, there is no way to improve the error performance without redundant sources of data. Here, it is worth noting that once the global information is available to entire PLUG network, mesh relaxation [8] and many other techniques can be used to improve the error propagation.

However, the author's localization algorithm is intended for the long-term deployment. Thus, the best hope for better performance is when pair-wise distance estimation converges to the true distance. Once the pair-wise distance estimations are accurate enough, the PLUG localization system can localize the PLUGs, even with correct rotation and transposition.



(a) PLUG 1 and 2 initiliazed

(b) PLUG 3's location is estimated

(c) PLUG 4's location is estimated

(d) PLUG 6's location is estimated

(e) PLUG 5's location is estimated

(f) PLUG 7's location is estimated

Figure 6-5: Steps for simulated active localization: **(a)** PLUG 1 and PLUG 2 are reference points. Thus, these two points are already given even before the localization algorithm starts. **(b)** Based on the positions of PLUG 1 and 2 and received distance information, PLUG 3's location is calculated. **(c)** Based on the positions of PLUG 1 and 3 and received distance information, PLUG 4's location is calculated. **(d)** Based on the positions of PLUG 3 and 4 and received distance information, PLUG 6's location is calculated. **(e)** Based on the positions of PLUG 1 and 4 and received distance information, PLUG 5's location is calculated. **(f)** Based on the positions of PLUG 2 and 5 and received distance information, PLUG 7's location is calculated.

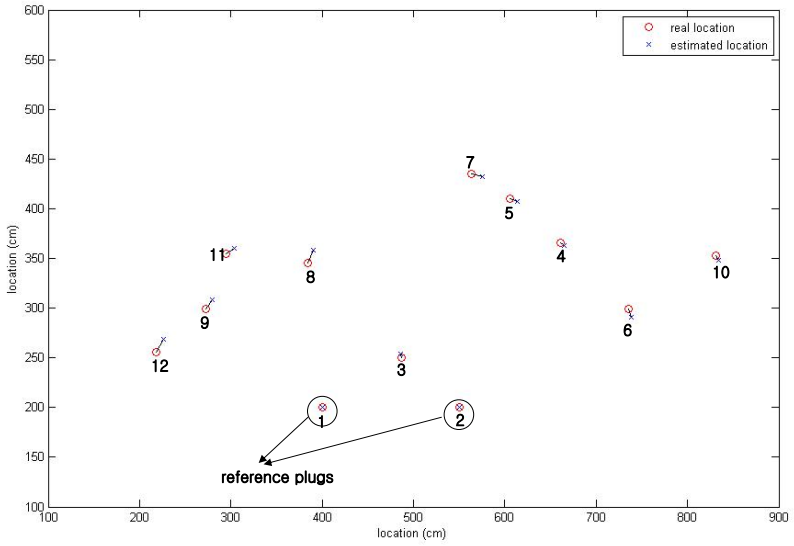


Figure 6-6: Error analysis plot for simulated active localization. The error is 7.97cm on average.

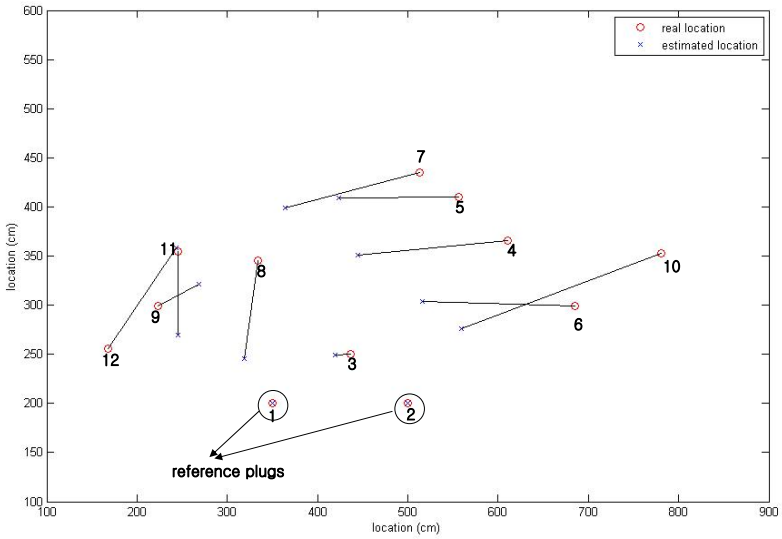


Figure 6-7: Error analysis plot for simulated passive localization. The error is 103.06cm on average.

6.4.3 Simulated passive localization

A similar method was used to simulate the passive localization. The error analysis is shown in Figure 6-7. The average error in the location, using the distance measurements from Table 6.2, is 103.06cm; this is worse than the error in simulated active localization. However, compared to the error in pair-wise distances from passive localization (50.96cm), the author believes that the location error is decent. This error is somewhat optimistic in simulation compared to the real deployment, where the real deployment estimates maximum time delay based on different types of acoustic stimuli that come from a variety of angles, not just in-line with the pair of nodes, as discussed in Section 5.6.2. Please note that, explained previously, the further the nodes are placed from the reference nodes, much larger error the estimation is.

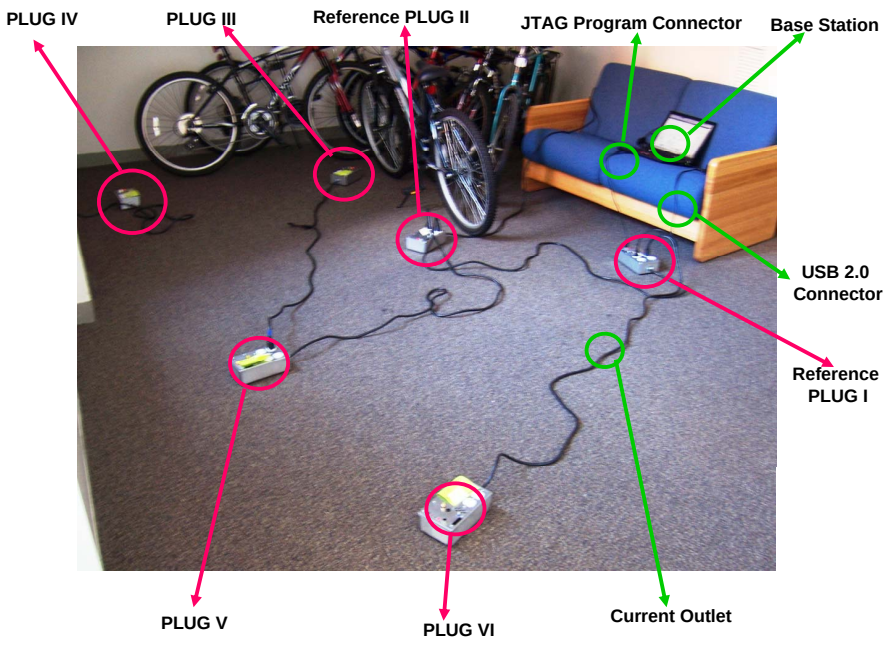


Figure 6-8: Deployment for active and passive localization test. This figure is annotated with base station, 2.0 USB connector for data collection, and 6 PLUGs connected each other for power. The PLUG5 was unlocalized due to its hardware failure.

6.5 Real Localization Results

For tests in a real deployment, 6 PLUGs were deployed in $7m$ by $5m$ dorm hallway as shown in Figure 6-8. The setup was not optimum for testing purposes. Due to its small size, sound reverberation and packet loss due to multi-path were prevalent. 6 PLUGs are positioned so that microphone, speaker and phototransistor face upward, the normal orientation of the PLUG platform. For performance comparison, active and passive localization are separately performed.

6.5.1 Active Localization

The active localization was not executable with the current system. If microphones and speakers do not face each other, the detection range with decent pair-wise distance estimation error is approximately $20cm$, which is poor performance considering that the size of PLUG is always $15cm$. Also, multi-path reflection off the walls and ceiling

would probably dominate over the direct PLUG-PLUG path. However, if the speaker and microphones are omni-directional, the author predicts that the active localization could perform as well as the simulation results show.

6.5.2 Passive Localization

On the contrary, the sound sources for passive localization tend to be omni-directional. Thus, the passive localization did not encounter the same problem that the active localization had. This localization responds to sounds such as door banging, pencil dropping, phone ringing, etc. The setup was untouched for approximately 20 hours while natural indoor sounds occurred. The ranges between pairs of nodes were estimated to be the non-outlier maximum as indicated in Section 5.6.2 and shown in Figure 6-13. This is expected in the absence of multi-path effects. The maximum delay corresponds to sound travelling directly from one node to the next with the source in-line with the node seperating, hence spanning their actual distance. Multi-path effect is also shown in the estimation error between node 1 and 2 in Figure 6-13; the error increase back up 14 hours after the deployment because multi-path distance is always longer than the actual distance.

Figure 6-10 shows $46.08cm$ error 10 minues after the deployment; Figure 6-11 shows $39.12cm$ error 1 hour and 36 minutes after the deployment; Figure 6-11 shows $20.3cm$ error 12 hours and 30 minutes after the deployment. The corresponding location estimation error for each node over 14 hours is shown in Figure 6-9. The PLUG 5 was unlocalized due to its hardware failure. As shown in both plots, the error improves as more data is available and this validates the author's algorithm.

It is worth noting following to make the plots more understadable. Because of the different locations of the PLUGs, all PLUGs did not receive the same number of triggers. The distance between node 1 and 2 is collected over 102178 collected samples. The distance between node 1 and 3 is collected over 54 samples. The distance between node 1 and 4 is collected over 160 samples. The distance between node 1 and 6 is collected over 217 samples. The distance between node 2 and 3 is collected over 102188 samples. The distance between node 2 and 4 is collected

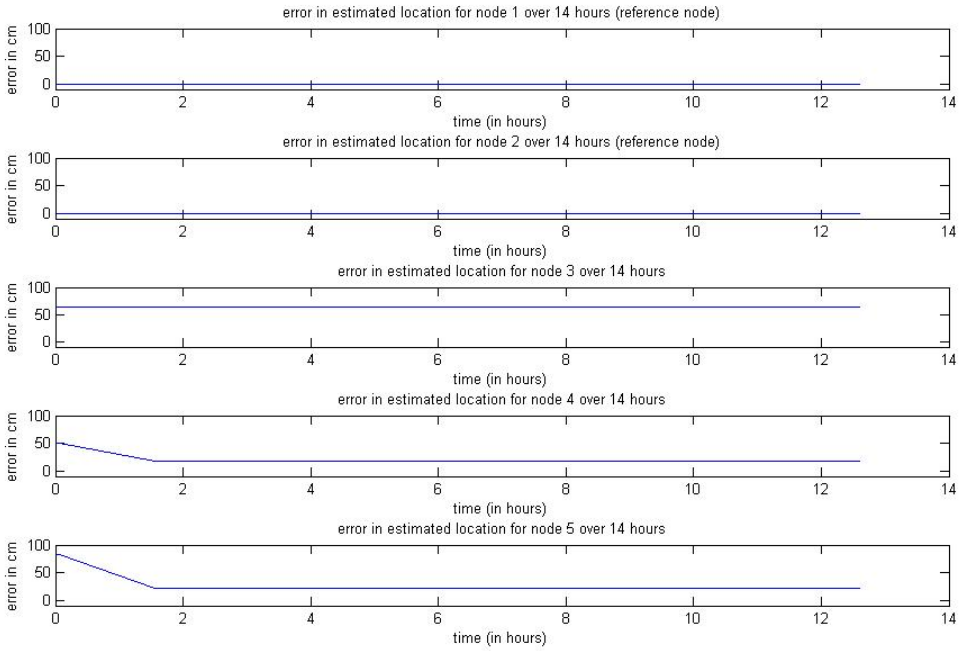


Figure 6-9: The error in location estimation using passive localization algorithm; the error changes over 20 hours.

over 102382 samples. The distance between node 2 and 6 is collected over 102370 samples. The distance between node 3 and 4 is collected over 255 samples. The distance between node 3 and 6 is collected over 210 samples. The distance between node 4 and 6 is collected over 431 samples.

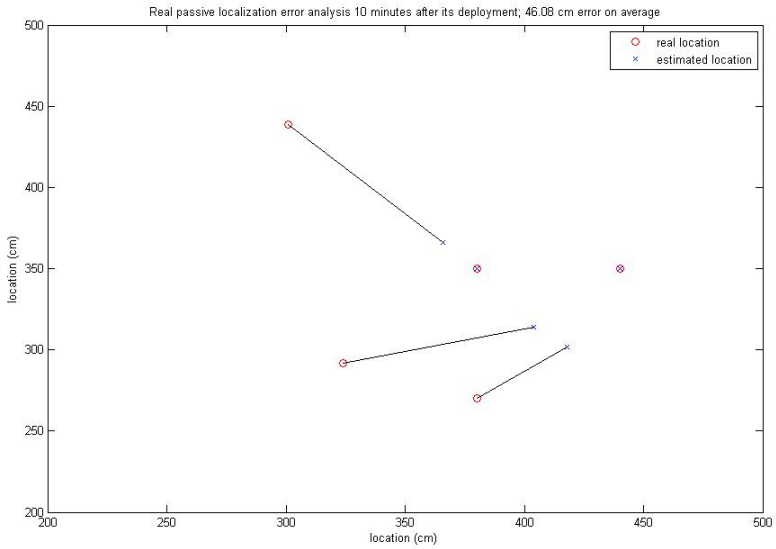


Figure 6-10: Error analysis for passive localition 10 minutes after the PLUGs have been deployed. It has 46.08cm error on average.

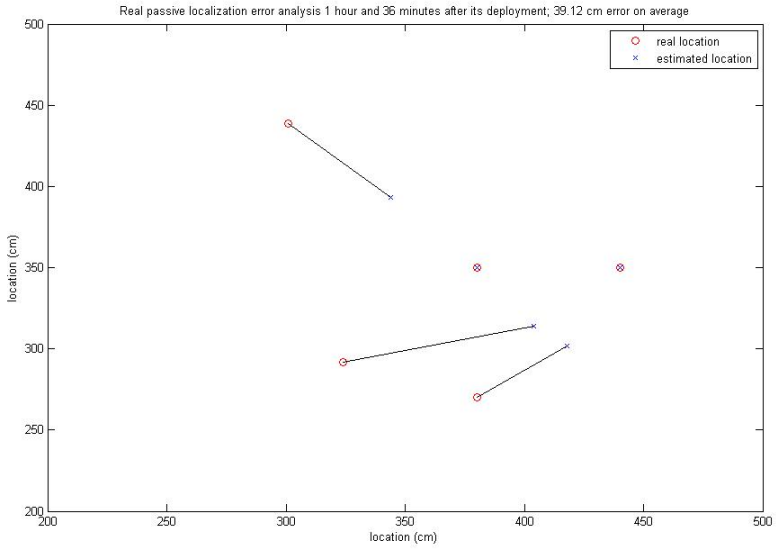


Figure 6-11: Error analysis for passive localition 1 hour and 36 minutes after the PLUGs have been deployed. It has 39.12cm error on average.

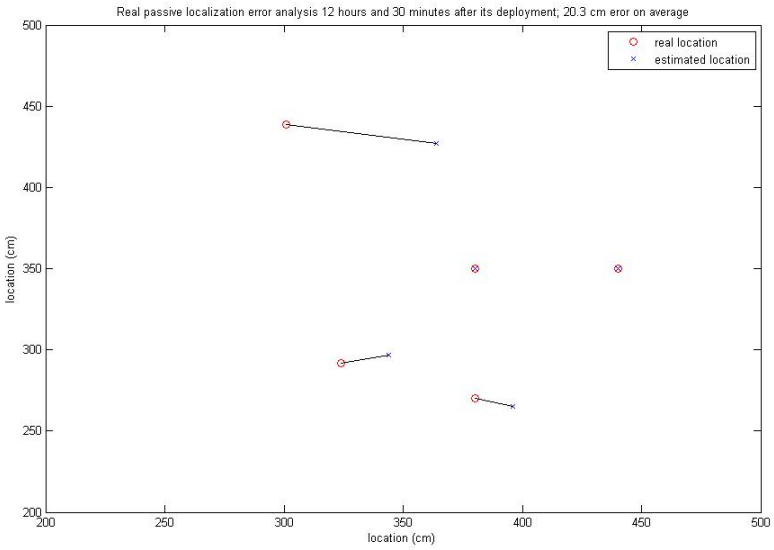
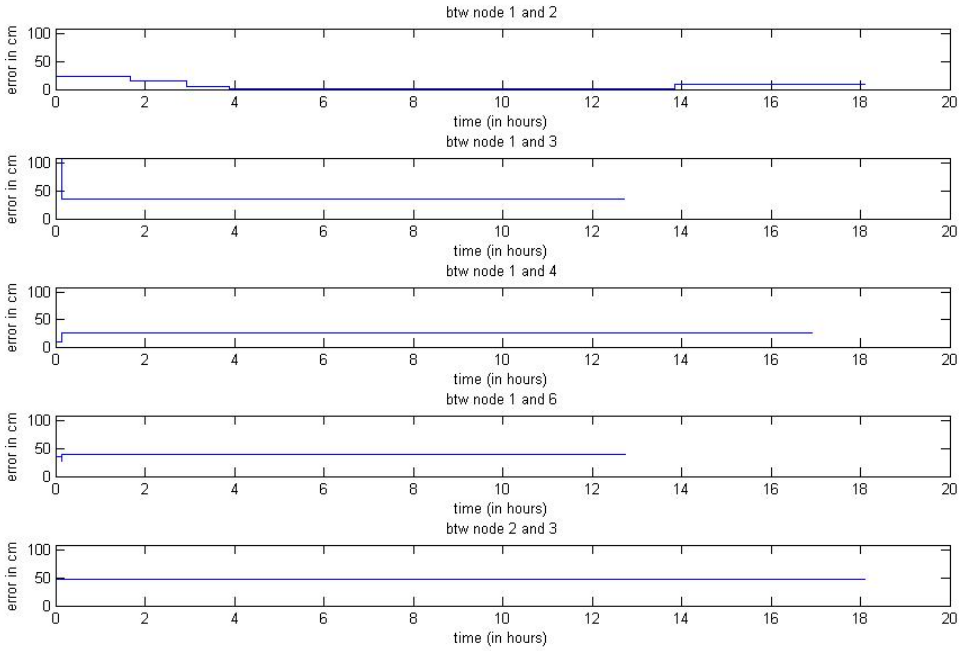
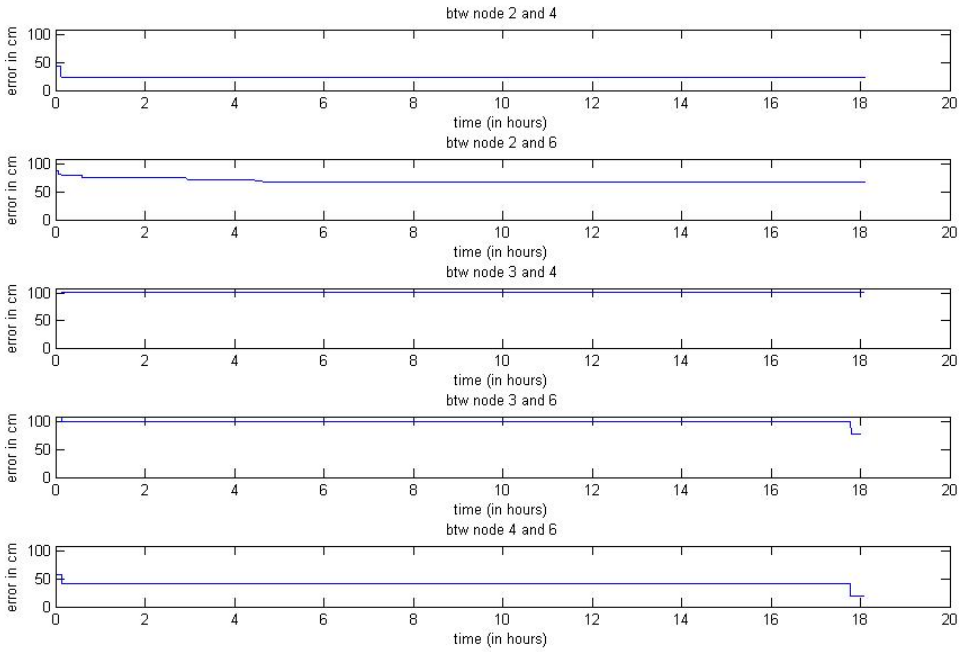


Figure 6-12: Error analysis for passive localition 12 hours and 30 minutes after the PLUGs have been deployed. It has 20.3cm error on average.



(a) For link between (node 1 and 2), (node 1 and 3), (node 1 and 4), (node 1 and 6), and (node 2 and 3).



(b) For link between (node 2 and 4), (node 2 and 6), (node 3 and 4), (node 3 and 6), and (node 4 and 6).

Figure 6-13: **(a) (b)** The error in pair-wise estimation based on the estimated time delay; the error changes over 20 hours.

Chapter 7

Closing Remarks

The goal of this thesis is to develop a distributed localization algorithm for the PLUG indoor sensor network by analyzing sound and light sensory data from naturally occurring phenomena. The author's approach has two main phases: passive and active. The system enters an active mode when its sensed region stays relatively silent and stable. Otherwise, it stays in the passive mode. In the passive mode, each node estimates its location based on sound sensory data and emulates the indoor environment based on light sensory data. In the active mode, each node estimates its location based on the occasionally generated mimics of natural phenomena, such as sonic transients (pencil dropping, coughing or water glasses clinking) or light source manipulation.

It is worth re-mentioning that, at least in the limited set of experiments that we ran, active ranging performs better than passive ranging, which performs somewhat better than ranging just using RSS. Active ranging had 2.1cm error on average, passive ranging had 50.96cm error on average, and ranging using RSS had 57.75cm error on average. For particular data with active ranging, simulated active localization achieved 2.98cm error on average, whereas the simulated localization using RSS ranging achieved 197.68cm error on average in radio silent environment. Simulated active localization achieved 7.97cm on average, whereas simulated passive localization achieved 103.06cm on average. Passive localization in a 20-hour long deployment showed that the error improves over time as more sensor data is available. In a test

with 6 PLUGs in $7m$ by $5m$ dorm hallway, we achieved passive localization errors of $46.08cm$, $39.12cm$, and $20.3cm$ when 10 mintues, 1 hour and 36 minutes, and 12 hours and 30 minutes passed after deployment respectively. Due to hardware limitations, active localization in real deployment was not executable. However, the author believes that modest hardware improvement will make it perform as well as the simulation for active localization. Although both ranging schemes and localization algorithms show good performance, there are still many possible extensions of the author's work as outlined below.

7.1 Future Work

Passive ranging can be improved by transmitting the entire sound wave and using the matched filter. However, the author worries that this might cause excessive network congestion. Currently, network packets with a maximum of 64-bytes can be sent at one time. The sound wave, recorded for 0.5 s in 4 KHz and 10 bits, has 0.5 s $\times$ 4000 Hz $\times$ 10 bits $\times$ $\frac{1 \text{ byte}}{8 \text{ bits}}$ = 2500 bytes size. This requires $\frac{2500}{64} = 40$ transmissions and receptions to send the entire 0.5 second long sound wave. If this is done by PLUGs in a neighborhood, this will certainly cause network congestion, so the author decided to rule this option out. However, this issue can be avoided in two ways. Problems from traffic can be minimized by adopting more sophisticated network schemes using TDMA, CDMA, or FDMA, or any combination of these, which the author believes is plausible. Then, transmitting the entire sound wave and using a matched filter to estimate the distance would greatly improve the passive ranging performance. The second option is transmitting a compressed version of the waveforms instead of entire sound wave. Vacher [88] showed that 7 wavelet coefficients (in a wavelet coefficients tree of 3 depth levels) can reasonably model sound transients and hence might improve the passive ranging accuracy.

In addition, because the PLUG is also equipped with a vibration sensor, it can also be useful to extract additional context; for example, different PLUGs can be assumed to be on the same surface if they detect the similar vibration behavior, just

like different PLUGs are assumed to be in the same room once they detect a similar light pattern. The entire localization system introduced in this thesis focuses on the room-scale localization. However, a similar localization method can be targeted for floor-scale use with simple modifications.

7.2 Conclusion

Technology Review [70] has picked sensor networks as one of top 10 emerging areas in its 2006 special report. Sensor networks have an enormous number of potential applications and more that are not even possible to list. However, there are still much research that needs to be done to make them robust and useful; localization especially indoors, is one of the issues. In this thesis, the author attempts to localize the PLUG sensor network and to draw coarse indoor environmental context (e.g. room borders) based on the sound and light sensory data collected from the PLUGs. However, there are many more interesting extensions of this work, and the author would like to invite the readers to explore those options.

Appendix A

Brief Technical Detail for Networked Cameras [25]

Funiak [25] formulates SLAT as a probabilistic inference task, where he maintains a joint distribution over possible `object_state`. `object_state` includes the object's velocity, location, and camera poses, given the images collected by the network. He focuses on two issues: algorithm's updatability and its scalability. First, let us talk about techniques used to update calibration. As often as it is in the field, representing, either exactly or approximately, unknown variables as Gaussian simplifies the problem. The author thus assumes unknown linear variables as Gaussian distribution and employs relative over-parameterization (ROP) and conditional hybrid linearization to approximate nonlinear variables as Gaussian. With a Gaussian representation of variables, ideas from Kalman filtering have been used to update variables with new observations. More precisely, the dynamic probabilistic SLAT process is divided into three phases: estimation, prediction, and roll-up. The estimation phase at time t is based solely on the previous `belief_state` at $t-1$ and current observation o^t. `belief_state` at time t represents updated estimations for variables, object's state and camera pose, based on observation up to time t.

Getting into a little more technical detail, let me briefly explain relative over-parameterization (ROP) and conditional hybrid linearization. Let's take a step back and think about what we are actually estimating. Although our goal is to estimate

camera pose, the object state needs to be estimated as well. ROP provides a way to represent both the camera pose and the object state. Object state is represented with two variables, M_t^x, x-coordinate at time t, and M_t^y, y-coordinate at time t, and these two variables are updated using following matrix operation:

$$M_t = \begin{bmatrix} M_t^x \\ M_t^y \\ M_t^{\dot{x}} \\ M_t^{\dot{y}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_{t-1}^x \\ M_{t-1}^y \\ M_{t-1}^{\dot{x}} \\ M_{t-1}^{\dot{y}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \epsilon \\ \epsilon \end{bmatrix}$$

where (M_t^x, M_t^y) is the object position and $(M_t^{\dot{x}}, M_t^{\dot{y}})$ is its velocity, and ϵ is a white noise variable giving additive noise in velocities. The camera pose is represented by three variables, u, v and ϕ. u is the distance from the object to its projection on the camera's image plane and v is the distance from this projection to the camera's center. As opposed to other conventional coordinate systems such as Cartesian coordinates or Polar coordinates of the camera and object, ROP makes it easier to approximate variables as Gaussians to apply Kalman filtering for the large camera network.

There is one other nonlinearity that needs to be approximated: the periodicity of the angle ϕ. In order to exploit the simple property of Gaussian variables, the periodicity needs to be eliminated. This problem is addressed in hybrid conditional linearization. Summing over all terms conditioned on several fixed angle (ϕ) values, we have a mixture of Gaussians. A mixture of Gaussian can be simplified to a single Gaussian by instantiated observation. This whole linearization process is called hybrid conditional linearization.

Appendix B

Alternate Quadratic Interpolation

Here, the author uses every data collected to find the best-fit quadratic algorithm, $y = ax^2 + bx + c$, by minimizing following value:

$$\sum_{i=1}^{k}(y_i - (ax_i^2 + bx_i + c))^2$$

where y_i and x_i for $i = \{1, 2, ..., k\}$ are cross-correlation values and its corresponding time stamp and k is the number of available data points (See Figure B-1 for illustration). In other words,

$$\{a, b, c\} = \arg_{a,b,c} \min \left[\sum_{i=1}^{k}(y_i - (ax_i^2 + bx_i + c))^2\right]$$

Thus,

$$\begin{aligned} a = \quad & \arg_a \tfrac{d}{da} \textstyle\sum_{i=1}^{k}(y_i - (ax_i^2 + bx_i + c))^2 = 0 \\ & \arg_a \left[a \textstyle\sum_{i=1}^{k} x_i^4 + \sum_{i=1}^{k}(bx_i^3 + (c - y_i)x_i^2)\right] = 0 \end{aligned}$$

Thus,

$$a = -\frac{\sum_{i=1}^{k}(bx_i^3 + (c - y_i)x_i^2)}{\sum_{i=1}^{k} x_i^4}$$

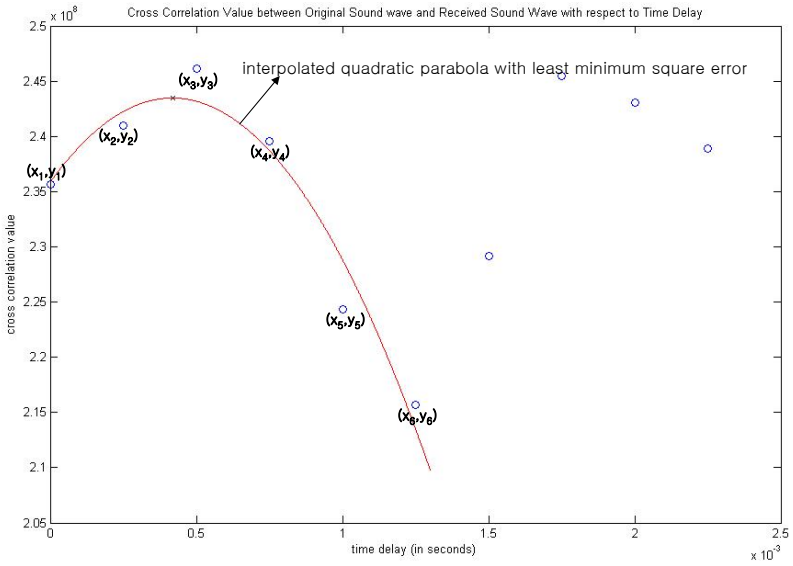


Figure B-1: Here, $k = 6$

Similarly,

$$\begin{aligned} b = & \quad -\frac{\sum_{i=1}^{k}(ax_i^3+(c-y_i)x_i)}{\sum_{i=1}^{k} x_i^2} \\ c = & \quad \frac{1}{k}\sum_{i=1}^{k}(y_i - ax_i^2 - bx_i \end{aligned}$$

Setting $p = \sum_{i=1}^{k} x_i$, $q = \sum_{i=1}^{k} x_i^2$, $r = \sum_{i=1}^{k} x_i^3$, $s = \sum_{i=1}^{k} x_i^4$, $m = \sum_{i=1}^{k} y_i x_i^2$, $n = \sum_{i=1}^{k} y_i x_i$, and, $l = \sum_{i=1}^{k} y_i$ and after simple manipulation, we have:

$$\begin{bmatrix} s & r & q \\ r & q & p \\ q & p & k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} m \\ n \\ l \end{bmatrix}$$

Rearranging, we have:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} s & r & q \\ r & q & p \\ q & p & k \end{bmatrix}^{-1} \begin{bmatrix} m \\ n \\ l \end{bmatrix}$$

From this, a and b are calculated. Just like the calculation in Chapter 5, the correlation is maximum at $-\frac{b}{2a}$ and the estimated distance is $-\frac{34800b}{4000\times 2a}$. As mentioned in Chapter 5, this scheme is mathematically more robust because it takes every collected sample into account for the estimation. However, this method does not perform better than the one presented in Chapter 5 because the estimation is sensitive to outliers; most of the structure in the correlation peak is defined by the few maximum points.

Bibliography

[1] Cesare Alippi, Alan Mottarella, and Giovanni Vanini. A rf map-based localization algorithm for indoor environments. *ISCAS: IEEE International Symposium on Circuits and Systems*, 2005.

[2] Atmel. *AT91SAM7S64 Data Sheet*. `http://www.atmel.com/dyn/resources/prod_documents/doc6175.pdf`.

[3] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. *INFOCOM*, 2000.

[4] BBC News. Tiny Wireless Memory Chip Debuts. `http://news.bbc.co.uk/2/hi/technology/5186650.stm`.

[5] A. Benbasat and J. Paradiso. Design of a real-time adaptive power optimal system. *Proceedings of IEEE in Sensors*, 2004.

[6] A. Benbasat and J. Paradiso. A compact modular wireless sensor platform. *IPSN*, pages 410–415, 2005.

[7] Jan Beutel. Geolocation in a picoradio environment. Master's thesis, UC Berkeley, 1999.

[8] Michael Joseph Broxton. Localization and sensing applications in the pushpin computing network. Master's thesis, MIT Media Laboratory, 2005.

[9] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, pages 28–34, 2000.

[10] Nirupama Bulusu and Sanjay Jha. *Wireless Sensor Network Systems*. Artech House, Boston and London, 2005.

[11] William Butera. *Programming a Paintable Computer*. PhD thesis, MIT Media Laboratory, 2002.

[12] James Caffery and Gordon Stuber. Overview of radiolocation in cdma cellular systems. *IEEE Communications Magazine*, 1999.

[13] K. Chakrabarty and S.S.Lyengar. *Scalable Infrastructure for Distributed Sensor Networks*. Springer, London, 2006.

[14] CHIPCON. *CC2500 Data Sheet*. `http://www.chipcon.com/files/CC2500_Data_Sheet_1_2.pdf`.

[15] Benjamin Christopher Dalton. Audio-based localization for ubiquitous sensor networks. Master's thesis, MIT Media Laboratory, 2005.

[16] Chrostoam Decker, Michael Beigl, Adam Eames, and Uwe Kubach. Digiclip: Activating physical documents. *24th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2004.

[17] Eiman Elnahrawy, Xiaoyan Li, and Richard P. Martin. The limits of localization using signal strength: A comparative study. *SECON: Sensor and Ad Hoc Communications and Networks*, 2004.

[18] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *Proceedings of the 5th symposium on Operating systems design and implementation*, 2002.

[19] Emre Ertin, Anish Arora, Rajiv Ramnath, Vinayak Naik, Sandip Bapat, Vinod Kulathumani, Mukundan Sridharan, Hongwei Zhang, Hui Cao, and Mikhail Nesterenko. Kansei: a testbed for sensing at scale. *IPSN*, pages 399–406, 2006.

[20] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordinates in sensor networks. *Mobile Computing and Networking*, 1999.

[21] FAIRCHILD SEMICONDUCTOR. *MOC3023 Data Sheet*. `http://www.ortodoxism.ro/datasheets/fairchildMOC3023-M.pdf`.

[22] Federal Communication Commission (FCC). 911 Services. `http://www.fcc.gov/911/`.

[23] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Mobile agent middleware for sensor networks: An application case study. *IPSN*, 2005.

[24] W. Foy. Position-location solution by taylor series estimation. *IEEE Transactions of Aerospace and Electronic Systems*, 12(2):187–193, 1976.

[25] Stanislav Funiak, Carlos Guestrin, Mark Paskin, and Rahul Sukthankar. Distributed localization of networked cameras. *IPSN*, 2006.

[26] Craig Gotsman and Yehuda Koren. Distributed graph layout for sensor networks. *Proceedings of the International Symposium on Graph Drawing*, 2004.

[27] Andy Harter and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[28] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. *ACM/IEEE MOBICOM*, 1999.

[29] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. *MOBICOM*, 2003.

[30] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 2001.

[31] Andrew Howard, Maja J Mataric, and Gaurav Sukhatme. Relaxation on a mesh: a formalism for generalized localization. *IROS (International Conference on Intelligent Robots and Systems)*, 2001.

[32] W.W. Irving and J.N. Tsitsiklis. Some properties of optimal thresholds in decentralized detection. *IEEE Transactions on Automatic Control*, 39(4):835–838, 1994.

[33] Paul Goud Jr., Christian Schlegel, Witold A. Krzymien, and Robert Hang. Multiple antenna communications systems: An emerging technology. *Canadian Journal of Electrical and Computer Engineering: Special Issue on Advances in Wireless Communications and Networking*, 29:51–59, 2004.

[34] Oliver Kasten and Kay Romer. Beyond event handlers: Programming wireless sensors with attributed state machines. *IPSN*, 2005.

[35] K.M.Hall. An r-dimensional quadratic placement algorithm. *Management Science 17*, pages 219–229, 1970.

[36] Yehuda Koren. On spectral graph drawing. *Proceedings of the 9th International Computing and Combinatorics Conference*, 2003.

[37] Branislav Kusy, Akos Ledeczi, Miklos Maroti, and Lambert Meertens. Node density independent localization. *IPSN*, pages 441–448, 2006.

[38] M. Laibowitz and J. Paradiso. Parasitic mobility for pervasive sensor networks. *PERVASIVE*, pages 255–278, 2005.

[39] L Lamport. Time clocks and the ordering of events in a distributed system. *ACM*, pages 558–564, 1978.

[40] T. E. Landers and R. T. Lacoss. Long range acoustic tracking of low flying aircraft. *Distributed Sensor Nets Sponsored by DARPA and Information Processing Techniques Office*, pages 68–77, 1978.

[41] Akos Ledeczi, Peter Volgyesi, Miklos Maroti, Gyula Simon, Gyorgy Balogh, Andras Nadas, Branislav Kusy, Sebestyen Dora, and Gabor Pap. Multiple simultaneous acoustic source localization in urban terrain. *IPSN*, 2005.

[42] Philip Levis and David Culler. Mate: A tiny virtual machine for sensor networks. *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[43] Philip Levis, David Gay, and David Culler. Bridging the gap: Programming sensor networks with application specific virtual machines. *U.C. Berkeley Technical Report*, 2004.

[44] Joshua Lifton, Deva Seetharam, Michael Broxton, and Joseph Paradiso. Pushpin computing system overview: A platform for distributed, embedded, ubiquitous sensor networks. *Pervasive Computing*, 2002.

[45] Rowley Associates Limited. *CrossWorks For ARM Manual*. `http://www.rowley.co.uk/arm/arm_manual.pdf`.

[46] Andrew Lippman and David P. Reed. Viral communications. *Media Laboratory Research*, 2003.

[47] Konrad Lorincz, David Malan, Thaddeus Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *Pervasive Computing*, 2004.

[48] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. *International Conference on Management of Data*, 2003.

[49] Samuel Madden, Robert Szewczyk, Michael Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. *Mobile Computing Systems and Applications*, 2002.

[50] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. *WSNA (Wireless Sensor Networks and Applications)*, 2002.

[51] Miklos Maroti, Branislav Kusy, Gyula Simon, and Akos Ledeczi. The flooding time synchronization protocol. *SenSys (Conference on Embedded Networked Sensor Systems)*, 2004.

[52] Mathew Laibowite and Joseph Paradiso. The UbER-Badge. `http://www.media.mit.edu/resenv/badge/index.html`.

[53] Mobile Megazine. Global Positioning System (GPS). `http://www.mobilemag.com/content/100/350/C5780/`.

[54] R. Nagpal. Organizing a global coordinate system from local information on an amorphous computer. *A.I. Memo 1666, MIT A.I. Laboratory*, 1999.

[55] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. *IPSN*, 2003.

[56] Ryan Newton and Matt Welsh. Region streams: Functional macroprogramming for sensor networks. *International Workshop on Data Management for Sensor Networks (DMSN)*, 2004.

[57] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. Landmarc: Indoor location sensing using active rfid. *Pervasive Computing and Communications*, 2003.

[58] D. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Telecommunication Systems*, 2003.

[59] M. Niedermayer, S. Guttowski, R. Thomasius, D. Polityko, K. Schrank, and H.Reichl. Miniaturization platform for wireless sensor nodes based on 3d-packaging technologies. *IPSN/SPOTS*, pages 391–398, 2006.

[60] Brendan O'Flynn, S. Bellis, K. Delaney, J. Barton, S.O'Mathuna, Andre Barroso, J. Benson, U. Roedig, and C. Sreenan. The development of a novel miniaturized modular platform for wireless sensor networks. *IPSN/SPOTS*, pages 370–375, 2005.

[61] O.G.Morchon, H. Baldus, and D.S.Sanchez. Resource-efficient security for medical body sensor networks. *BSN*, 2006.

[62] Mark Paskin, Carlos Guestrin, and Jim McFadden. A robust architectur for distributed inference in sensor networks. *IPSN*, 2005.

[63] Adrian Perring, Robert Szewczyk, Victor Wen, David Culler, and J.D. Tygar. Spins: Security protocols for sensor networks. *Mobile Computing and Networking*, 2001.

[64] G. Polychronopoulos and J.N. Tsitsiklis. Explicit solutions for some simple decentralized detection problems. *IEEE Transactions on Aerospace and Electronic Systems*, 26:282–291, 1990.

[65] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. *MIT LCS Technical Report*, 892, 2003.

[66] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. *Proceedings of International Conference on Mobile Computing and Networking*, pages 32–43, 2000.

[67] Theodore S. Rappaport. *Wireless Communications: Priciples and Practice*. Prentice Hall, 2002.

[68] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. *Wireless Communications and Networking Conference (WCNC)*, 2005.

[69] R. Sarpeshkar. Ultra low power electronics for medicine. *BSN*, 2006.

[70] Neil Savage. 10 emerging technologies: Pervasive wireless. *Technology Review*, 2006.

[71] Andreas Savvides, W. Barger, S. Adlakha, R. Moses, and M.B.Srivastava. On the error characteristics of multihop node localization in ad-hoc sensor networks. *IPSN*, pages 567–577, 2003.

[72] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. *ACM SIGMOBILE*, 2001.

[73] Andreas Savvides, Heemin Park, and Mani B. Srivastava. The n-hop multilateration primitive for node localization problems. *First ACM International Workshop on Wireless Sensor Networks and Application*, 2003.

[74] Seongju Chang. Research Highlights: Smart Architectural Surfaces. `http://ttt.media.mit.edu/research/sas.html`.

[75] World's Smallest and thinnest. Hitachi. `http://www.hitachi.com/New/cnews/060206.html`.

[76] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. Tracking moving devices with the cricket location system. *USENIX/ACM MOBISYS*, 2004.

[77] A. Soppera and T. Burbridge. Maintaing privacy in pervasive computing-enabling acceptance of sensor-based services. *BT Technology Journal*, 22(3):106–118, 2004.

[78] R. Stoleru and J.A.Stankovic. Probability grid: a location estimation scheme for wireless sensor networks. *SECON*, pages 430–438, 2004.

[79] Radu Stoleru, Tian He, John Stankovic, and David Luebke. A high-accuracy, low-cost localization system for wireless sensor networks. *SenSys*, pages 13–26, 2005.

[80] Horst Stormer. 21 Ideas for the 21st Century. `http://www.businessweek.com/1999/99_35/b3644001.htm`.

[81] W. P. Tay, J. N. Tsitsiklis, and M. Z. Win. Asymtotic performance of a censoring sensor network. 2006.

[82] Christopher Taylor, Ali Rahimi, Jonathan Bachrach, and Howard Shrobe. Simultaneous localization and tracking in an ad hoc sensor network. *IPSN*, 2006.

[83] Sebastian Thrun. Affine structure from sound. 2005.

[84] J.N. Tsitsiklis. Decentralized detection by a large number of sensors. *Mathematics of Control, Signals and Systems*, 1(2):167–182, 1988.

[85] J.N. Tsitsiklis. Extremal properties of likelihood-ratio quantizers. *IEEE Transactions on Communications*, 41(4):550–558, 1993.

[86] J.N. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, 30(5):440–446, 1985.

[87] G. Turing, W. Jewell, and T. Johnston. Simulation of urban vehicle-monitoring systems. *IEEE Transactions on Vehicular Tehnology*, 21(1):9–16, 1972.

[88] Michel Vacher, Dan Istrate, and Jean-Francois Serignat. Sound detection and classification through transient models using wavelet coefficient trees. *Proc. 12th European Signal Processing Conference*, pages 367–372, 2004.

[89] M. Weiser. The computer for the 21st century. *Pervasive Computing*, pages 18–25, 2002.

[90] Rob Weiss. *Cellular Computation and Communications using Engineered Genetic Regulatory Networks*. PhD thesis, MIT, 2001.

[91] G. Werner-Allen, J. Johnson amd M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. *EWSN*, 2005.

[92] WhereNet. Company Homepage. http://www.wherenet.com/.

[93] Kamin Whitehouse and David Culler. Macro-calibration in sensor/actuator networks. *Mobile Networks and Applications 8*, 8:463–472, 2003.

[94] Wikipedia. Global Positioning System (GPS). http://en.wikipedia.org/wiki/GPS.

[95] Wikipedia. LORAN. http://en.wikipedia.org/wiki/LORAN.

[96] Wikipedia. Sensor Web. http://en.wikipedia.org/wiki/Sensor_Web.

[97] A Pervasive 2005 Workshop. What makes for good application-led research in ubiquitous computing? *Pervasive*, 2005.

[98] Christopher Wren and Srinivasa Rao. Self-configuring, lightweight sensor networks for ubiquitous computing. *Ubiquitous Computing*, 2003.

[99] Shunzo Yamashita, Takanori Shimura, Kiyoshi Aiki, Koji Ara, Yuji Ogata, Isamu Shimokawa, Takeshi Tanaka, Hiroyuki Kuriyama, Kazuyuki Shimada, and Kazuo Yano. 15 mm, 1μa, reliable sensor-net module: enabling application-specific nodes. *IPSN/SPOTS*, pages 383–390, 2006.

[100] Kiran Yedavalli, Bhaskar Krishnamachari, Sharmila Ravula, and Bhaskar Srinivasan. Ecolocation: A sequence based technique for rf localization in wireless sensor networks. *IPSN*, 2005.