An RSSI-based Error Correction Applied to Estimated Sensor Locations

Masashi Sakurada
Graduate School of Science and Engineering
Ehime University
Matsuyama, Ehime 790-8677, Japan

Munehiro Fukuda
Computing & Software Systems
University of Washington, Bothell
18115 NE Campus Way, Bothell, WA 98011
mfukuda@uw.edu

***Abstract*—We consider a wireless sensor network (WSN) that needs to install hundreds of sensor nodes over a vast extent of land, (e.g., an orchard temperature sensor network). For monitoring critical events such as frost-prone temperature over an orchard, WSN users, (namely farmers) need to identify all sensor locations, which is however overwhelming from the viewpoints of costs and man power. To address this problem, we enabled non-GPS-adapted sensors to accurately estimate their locations by applying our RSSI (received signal strength indication)-based error-correcting algorithm to a conventional location-estimating method named Gomashio [1]. The paper presents our algorithm and demonstrates its performance improvement in estimating sensor locations as compared to Gomashio.**

I. Introduction

For the last three years we have been developing an agent-based workbench for on-the-fly sensor-data analysis that particularly focuses on providing crop growers with on-going and near-future air-temperature information about their orchard for frost-danger prediction [2]. This workbench collects temperature data from a crop grower's WSN and applies them to temperature-predicting programs [3], [4], [5] that we have parallelized with our multi-agent spatial simulation (MASS) library [6].

For our system implementation and verification, we have collaborated with a crop grower and two local WSN companies in Washington State, U.S.A. Through our collaboration and discussion, we estimated that the WSN configuration summarized in Table I would facilitate a detailed temperature-monitoring environment to our crop grower. More specifically, we assume that an orchard with a 120-acre space installs 25 Viking X sensors, each operated at 900MHz, adapted with a GPS, and arranged with 8 non-GPS-adapted ZigBee sensors, which thus brings the total number of ZigBees to 200. ZigBee sensors capture and send their local climate information such as temperature, moisture, and possibly image data to the near-by Viking X that then forwards to the central sink node. We also assume that each sensor should work as an autonomous node in ad-hoc communication. Of importance is how to locate the positions of all the 200 Zigbee sensors.

An orchard area	120 acres
The number of Valhalla Wireless' Viking Xs [7]	25
The number of AgComm's ZigBee sensors [8]	200
Radio propagation radius	60m

TABLE I. Orchard parameters

The simplest way to locate these sensors is to manually take notes of their locations identified with a GPS device when installing the sensors over an orchard. However, it is overwhelming to perform this task for 200 sensor nodes. Furthermore, it is also economically impractical to adapt a GPS device to all these ZigBee sensors for just obtaining their location information. Therefore, we should estimate their locations, using software that runs on these sensor nodes. Because most sensor nodes are battery-powered, such a program that estimates sensor locations should be computationally light. We also anticipate that there are many obstacles in an orchard such as high trees and wind generators. Therefore, location-identifying algorithms should not be affected by obstructions. In summary, the following list shows requirements for identifying sensor node locations:

- High accuracy for location identification
- No special hardware (such as GPS).
- Light calculation
- Less susceptible to obstruction

To address the above requirements, we started our design with a conventional location-identifying algorithm named Gomashio [1]. In this algorithm, each sensor node identifies its own location, using its communication radius and hop counts from an anchor node, (i.e., a node that has already identified its own location). Therefore, it does not require any special hardware, is not affected by obstacles, and can lower the amount of calculation as compared to the SOM algorithm [9] that uses competitive learning. However, Gomashio has a disadvantage to accumulate errors in subsequently calculating a new sensor location, based on anchor node's estimated location (which may already have some errors).

To minimize such errors in location identification, we first estimate sensor locations with Gomashio, and thereafter correct accumulated errors, using our RSSI-ranging algorithm. This improves the accuracy in estimating sensor locations. This paper presents our RSSI-based error-correcting algorithm and demonstrates its accurate error correction as compared to Gomashio.

II. Related Work

Among many algorithms that estimate sensor locations, we examine the following four: (1) RADAR [10] and WiPS [11], (2) Centroid [12], (3) SOM [9], and (4) Gomashio [1], all satisfying the four requirements that we defined in Section I. Thereafter, we focus on Gomashio as the best choice to be used for our target orchard model and discuss its technical issues we still have to address.

A. RADAR and WiPS

RADAR [10] and WiPS (Wireless LAN based Indoor Positioning System) [11] estimate the distance between two sensor nodes with RSSI, (i.e., one of the parameters used in WSNs), and identify their locations, using trilateration. As shown below, they obtain the distance information with the inverse function of the free space propagation loss model (FPSL).

$$\begin{aligned} P(d) &= P_0 - 20\log_{10}(\frac{4\pi d}{\lambda})[dBm] \qquad (1) \\ where & \\ P_0 &= RSSI\ at\ distance\ 0[m] \\ \lambda &= \frac{c}{f} = \frac{3 \times 10^8[m/s]}{2.4[GHz]} \end{aligned}$$

WiPS allows neighboring WSN nodes to measure a distance to each other. This method can increase the amount of location-identifying data and therefor improve the accuracy in positioning WSN nodes. However, RSSI is susceptible to various environments such as shadowing, diffraction, and multi-path fading. Therefore, it is not always possible to estimate accurate location only with RSSI.

B. Centroid

In Centroid [12], a WSN node that needs to identify its location computes the average coordinate position of all anchor nodes visible to it, and accepts this value as its location. More specifically, given N anchor nodes, a node estimates its position by calculating the center of mass with the following formula:

$$(X, Y) = (\frac{X_1 + \cdots + X_N}{N}, \frac{Y_1 + \cdots + Y_N}{N}) \qquad (2)$$

However, an actual node location is not always coincident with the center of mass among all the anchor nodes in the coordinate system. Therefore, the accuracy of Centroid's location identification is questionable.

C. SOM

SOM (Self-Organizing Map) [9] estimates a sensor position, using competitive learning. Competitive learning is based on neural network whose multiple neurons react to a single input data item. In this machine-learning model, only the neuron with the strongest output will survive to receive a larger weight for the network connection than the other neurons. SOM first gives each network node a random location. Thereafter, each node exchanges its locational information including its updated position as well as the number of hops to and the distance from neighboring nodes, and updates its position subsequently so as to satisfy conditions given from neighbors. Through a repetition of this learning phase, SOM can get its estimation closer to an actual network topology.

SOM has merits in reducing the number of anchor nodes to be used for its topology estimation and in estimating node positions with high accuracy, whereas it has demerits in exhanging a large number of packets. This means the more accuracy, the more learning phases that increase network traffic.

D. Gomashio

Gomashio [1] calculates a highly promising area that may include a target node to be identified around anchor nodes, using the minimum number of hops from each anchor to the target and each anchor's communication radius. When a target node knows the number of hops from each of multiple anchors, it can identify the area overlapped with circles, each corresponding to a different anchor's communication range. This area is then considered to include the target node. If the area is smaller than a given threshold value, the target node is promoted to a new anchor node and is applied to positining other sensor nodes. In the following, we give more details on how to determine a range that includes a given target node to be identified. The explanation below assumes that a target node maintains its anchor node information as shown in Table II.

Anchor node	Location	# hops
G_a	(X_a, Y_a)	2
G_b	(X_b, Y_b)	2
G_c	(X_c, Y_c)	2

TABLE II. ANCHOR NODE INFORMATION

1) For each anchor, the target node determines the circle that centers this anchor node and has a radius with a value obtained by multiplying the anchor's communication range by the hop count from the anchor to the target. (See Figure 1-(a)).
2) For each circle obtained in phase 1, the target node determines the smallest square that encompasses this circle. Thereafter, the target node identifies the rectangle that is overlapped with all these squares. (See Figure 1-(b)).
3) To determine if the target node can exist in the rectangle obtained in phase 2 with a high probability, the target applies all the intra-rectangle's coordinates (x_i, y_i) to the following formulae. In the formulae, let h_g be the hop count from anchor node, r be the communication radius, (x_g, y_g) be the position coordinate of an anchor node, and n be the number of anchor nodes known to the target node.
If $h_g = 1$:
$$(x_i - x_g)^2 + (y_i - y_g)^2 < r^2$$
If $h_g > 1$:
$$(x_i - x_g)^2 + (y_i - y_g)^2 < r^2 h_g^2$$
$$(x_i - x_g)^2 + (y_i - y_g)^2 > r^2$$
For a given coordinate (x_i, y_i), if all (x_g, y_g) (where $0 < g < n$) meets these formulae, (x_i, y_i) is considered as one of the target node's possible coordinates.

After phase 3, the target node chooses a set of minimum and maximum coordinates (x_i, y_i), from which it identifies the final rectangle that includes the target, as shown in Figure 1-(c). Finally, the target node then identifies this rectangle's centroid as its estimated position.

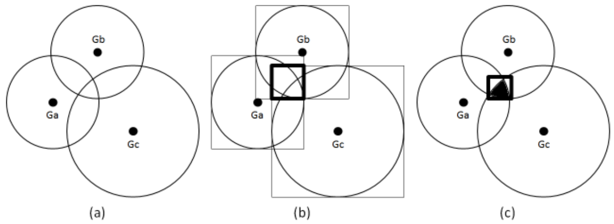

Fig. 1. A rectangulation-based algorithm for locating a sensor

Anchor node	Location	Hop
Ga	(Xa,Ya)	1
Gb	(Xb,Yb)	1
Gc	(Xc,Yc)	1

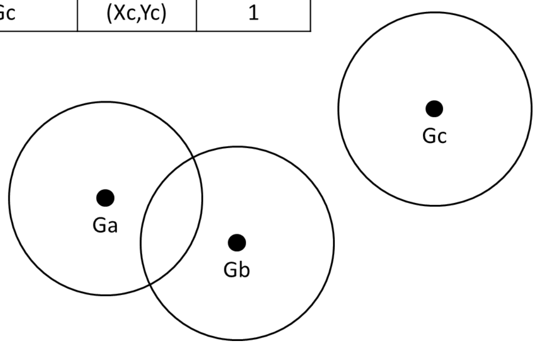

Fig. 2. A false example of rectangulation

Gomashio uses an estimated node as a new anchor node that will be used in the following estimation. Therefore, if there is an error in an estimated position, the error will be accumulated to the next node's estimation. This repetitive location identification inevitably deteriorates the accuracy of estimating network-wide node location.

E. Technical Issues

Gomashio uses each anchor node's communication radius and the smallest number of hops from a target to the anchor node. The estimated node is promoted to a new anchor node that is then applied to the next location identification. Therefore, subsequent calculations may amplify estimation errors large enough to make the entire node estimation inaccurate. This inaccuracy may cause a contradiction exemplified in Figure 2.

Figure 2 indicates that a target node maintains its anchor node information in the table showing its reachability to all three anchors within 1 hop. However, anchor node G_c's coordinate is actually outside of the communication radius. This contradiction is brought by errors that are frequently accumulated through Gomashio's repetitive location estimation. This abnormality prevents Gomashio from calculating an overlapped rectangle as shown in Figure 1-(b).

III. Proposed Algorithm

When using a WSN to monitor the orchard climate, we need to consider that obstacles in an orchard, (e.g., high trees and wind generators) put noises on RSSI and thus add errors to RSSI-based sensor location estimation such as

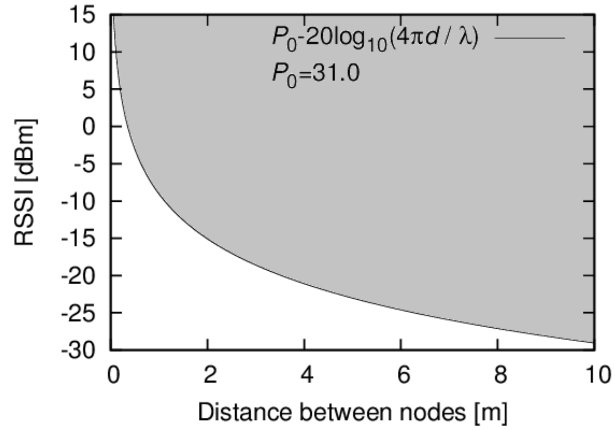

Fig. 3. Relationship between RSSI and inter-node distance

RADAR [10] and WiPS [11]. As discussed in the previous section, SOM [9] requires a substantial amount of computation that is not suitable to battery-operated sensors. Therefore, we identify sensor locations, using Gomashio that is based on the number of hops to and communication radius of each anchor node for mitigating noises brought by obstacles. To even correct Gomashio's accumulated errors (which we discussed with Figure 2), we calculate the distance between two sensor nodes, using the radio wave propagation model that shows the correlation between the RSSI strength and the distance from a given radio, and adjust the Gomashio-estimated node locations with our calculation. The more details of our error-collecting algorithm are given as follows.

A. Location-Error Correction Algorithm

Figure 3 draws a graph of Formula 1 that represents FPSL. Our proposed algorithm is based on FPSL to calculate the distance from a sender to a receiver radio by observing the sender's RSSI strength at the receiver. We then apply the calculated distance to Gomashio-based estimation of these two radio locations.

In most cases, radio propagation errors occur toward a direction to attenuate RSSI. This in turn means that we are unlikely to encounter contradictory cases that have both a strong RSSI and a long distance from a radio-signal sender, which corresponds to the shaded portion (or a gray-colored portion) in Figure 3. We use this RSSI-distance correlation in order to correct Gomashio's accumulated errors. The following three steps are the essence of our error-correcting algorithm that will be applied to Gomashio-estimated sensor locations. Note that d_gij is the distance between anchor nodes i and j obtained by the Gomashio calculation; d_rij is the distance between anchor node i and j obtained by RSSI; and R is each anchor node's communication radius. We assume that anchor node j is located within the communication radius of anchor node i. We apply the following three steps to all anchor nodes in a given network.

1) If $d_rij < d_gij$, we move anchor node j at the distance of d_rij from anchor i.
2) If $R < d_gij$, we move anchor node j at the distance distance of R from anchor node i.

3) If $d_r ij < R$, we move anchor node j at the distance of $d_r ij$ from anchor node i.

IV. Performance Evaluation

To demonstrate the efficiency of our error-collecting algorithm, we compared our algorithm with Gomashio in terms of accuracy of location identification, using simulation whose parameters are summarized in Table III.

Parameters	**Values**
Simulation space	$700 \times 700[m]$
Radio propagation radius	$60[m]$
# of ZigBee sensors	200
# of Viking X sensors	25
Node placement	a mesh network an irregular mesh network
Routing protocol	DSR
# of anchor nodes	$4, 5, 6, 7, 8, 9$, and 10
Existing range	$4000[m^2]$
RSSI maximum error	$20[dBm]$
Attenuation constant a	0.25
Simulation time	100

TABLE III. Simulation parameters

A. Simulator Design

The simulation assumes that a meshed WSN covers a $700m \times 700m$ space. The network installs 25 Viking X nodes, each surrounded by 8 ZigBee sensors in east, south, west, north, and diagonal directions, which thus brings the total number of sensor nodes to 225. Each ZigBee's communication radius is $60m$. We prepared two types of mesh networks: a regular mesh and a random mesh. The regular mesh places each sensor node exactly at a different interlaced position. The random mesh shifts each sensor node from an interlaced position randomly but a little enough to prevent any node from being isolated from the others, (i.e. being unable to communicate with the other nodes). The simulation assumes DSR (Dynamic Source Routing) that fits to an ad-hoc network environment, and counts the number of hops for a packet to travel from a source to a destination node. The threshold value of rectangular area to identify a node location (and thus to promote the node to a new anchor) is $4000m^2$. If this threshold value is small, we will obtain the higher accuracy of identifying a node location, however we may end up with a fewer nodes whose location can be estimated. On the other hand, a larger threshold value deteriorates the positioning accuracy. The threshold value of $4000m^2$ is slightly larger than $60m \times 60m$ that can obtain a high positioning accuracy and estimate a sufficient number of node locations. The number of anchor nodes used for estimating a new node position varies from 4 to 10. For each variation, we repeated 100 simulations.

Our experiment also simulated WSN communication based on the characteristics of radio wave propagation. The simulation generates RSSI, using FPSL that is represented by Formula 1. We also considered the influence of obstacles and added the corresponding errors to RSSI. Formula 3 models the function to add errors to RSSI, and Figure 4 draws the corresponding graph.

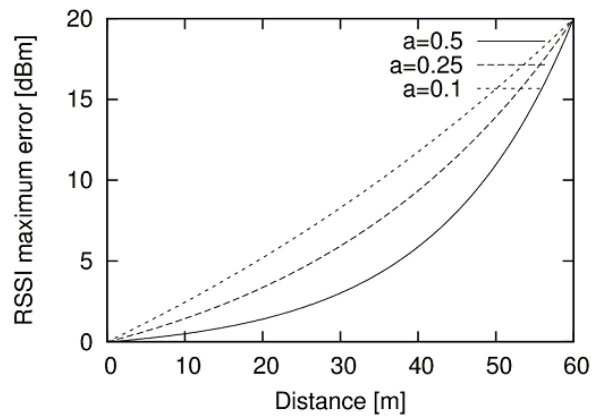

Fig. 4. The exponential growth of RSSI errors with distance

$$Err = f(dist) = 25 \times \frac{10^{\frac{a \times dist}{20}} - 1}{\delta} \qquad (3)$$

Here Err is the function that generates RSSI errors between two nodes in their $dist$ distance; a is an attenuation constant; and δ is an adjustment parameter that maintains Err in the range between 0 and 1. Note that δ is calculated from the following formula, assuming that $dist$ is at maximum $60m$.

$$\delta = 10^{\frac{a \times 60}{20}} - 1$$

We used random numbers to generate RSSI errors that are incurred by radio wave attenuation. Basically, the longer inter-node distance, the more RSSI errors. Assuming many obstacles in an orchard, we set the maximum RSSI errors to $20dBm$ when the distance between two nodes reaches the limit of their communication range ($60m$). This value was determined by our empirical data that was published in [13]. In this simulation, we considered the attenuation of radio waves due to the presence of obstacles but not the amplification of radio waves generated by wave interference. Therefore, in this simulation, RSSI errors work only in a negative direction.

Although Formula 3 defines ideal RSSI errors, the actual RSSI errors may vary and thus differ from simulation results. Therefore, we conducted our simulation with three different attenuation constants: 0.5, 0.25, and 0.1. We observed a similar trend in all simulation results. Therefore, the next section shows only the simulation results with attenuation constant of 0.25.

B. Simulation Results

We compared our RSSI-based error-correcting algorithm with Gomashio from the viewpoints of their error ranges and estimated network topologies.

Figure 5 shows the average positioning errors of Gomashio and our error-collecting algorithm (denoted as *Proposal* in the graph) when these two algorithms were applied to a regular mesh network. As shown in Figure 5, the positioning accuracy is improved as increasing the number of anchor nodes. The use of more anchor nodes narrows the rectangular area that

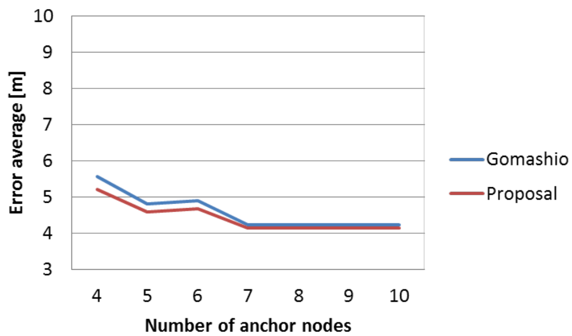

Fig. 5. The effect of error correction in a mesh network

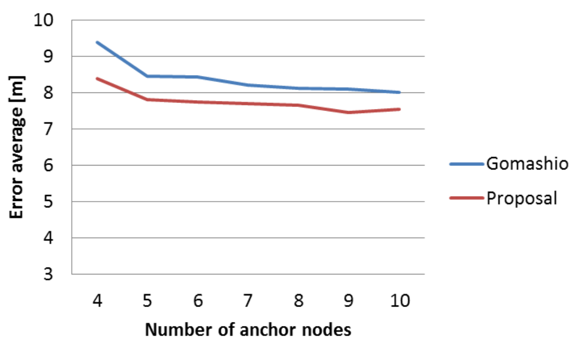

Fig. 6. The effect of error correction in an irregular mesh network

may include a target node to identify, which improves the accuracy of locating a target node. However, the positioning accuracy with 6 anchor nodes is slightly lower than that with 5 anchors. This was resulted from how anchor nodes were placed in the simulation space. Figure 5 demonstrates that our algorithm achieved higher accuracy than Gomashio in all cases where the number of anchors varied from 4 to 10. The average positioning errors of Gomashio was $4.61m$, whereas that of our error-collecting algorithm was $4.44m$.

Figure 6 compares the average positioning error of Gomashio and our error-correcting algorithm (denoted as *Proposal* in the graph) when these two algorithms were applied to a random mesh network. As shown in Figure 6, both the algorithms improved their positioning accuracy as increasing the number of anchor nodes. Our algorithm achieved better than Gomashio with any number of anchor nodes. The average positioning errors of Gomashio was $8.39m$, whereas our algorithm reduced positioning errors down to $7.76m$.

Figure 7 compares the original random mesh network, the Gomashio-estimated network, and our error-corrected network when we used 4 anchor nodes in both Gomashio and our error-corrected topology estimation. As shown in Figure 7, our algorithm retrieved a network closer to the original random mesh than Gomashio did. The average positioning error of Gomashio was $9.30m$, whereas that of our algorithm was $7.84m$.

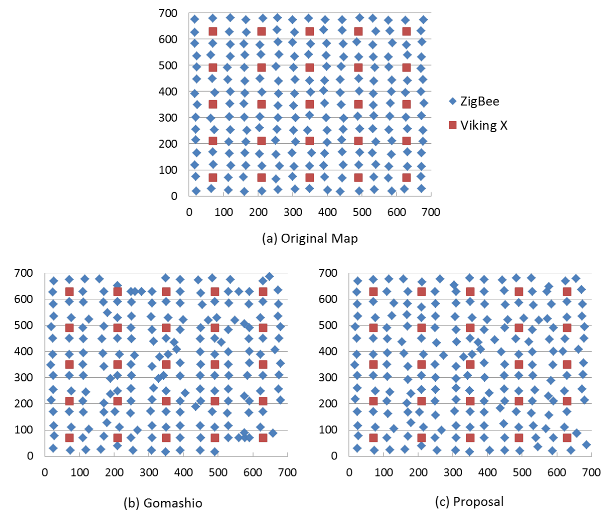

Fig. 7. A visualization of error correction in an irregular mesh network

V. Conclusions

This paper presented an RSSI-based error-correcting algorithm to be applied to estimated sensor locations, particularly focusing on identifying the position of 200 ZigBee sensor nodes that will be placed in a 120-acre orchard. We first adopted Gomashio to identifying sensor locations, and thereafter applied our error-correcting algorithm to estimated sensor locations. Our algorithm used the nature of radio propagation and facilitated more accurate sensor positioning.

Issues to address in the future are two-fold:

- Consideration of various communication radiuses
- Investigation and verification of radio wave propagation properties

We fixed each sensor node's communication radius to $60m$ in accordance with our target orchard environment. Needless to say, we should consider other communication radiuses than $60m$ for purpose of applying our error-collecting algorithm to different environments. Furthermore, we need to verify the effectiveness of our algorithm in various environments that allow a combination of multiple communication radiuses.

In this research, we simulated the characteristics of radio wave propagation, however we need to measure how much RSSI errors will occur in real orchards. Moreover, we assumed only the attenuation of radio waves. As a future plan, we will consider about shadowing and multi-path fading that fluctuate radio waves irregularly.

Acknowledgment

This research has been conducted under the UW Bothell and Ehime University academic exchange agreement.

References

[1] M. Iwatani, Y. Nishio, M. Murase, and H.Tokuda, "GOMASHIO: Node poistioning system in ad hoc snesor network," *IPSJ*, vol. Vol.2001, no. No.108, pp. 22–30, November 2001.

[2] M. Fukdua, “Agent-based workbench for on-the-fly sensor-data analysis,” in *In Proc. of 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing - PacRim 2011*, August 2011, pp. 333–339.

[3] B. A. Smith, G. Hoogenboom, and R. W. McClendon, “Artificial neural networks for automated year-round temperature prediction,” *Computers and Electronics in Agriculture*, vol. Vol.68, no. Issue.1, pp. 52–61, August 2009.

[4] UC Davis Biometeorology Program - Frost Protection, “http://biomet.ucdavis.edu/frost-protection.html.”

[5] Fred C. Collins Jr., “A Comparison of Spatial Interpolation Techniques in Temperature Estimation,” in *Third International Conference/Workshop on Integrating GIS and Environmental Modeling*. Santa Fe, NM: NCGIA, January 1996, pp. in CD–ROM.

[6] T. Chuang, “Design and qualitative/quantitative analysis of multi-agent spatial simulation library,” Master’s thesis, Master of Science in Computing and Software Systems, University of Washington, 2012.

[7] Valhalla Wireless, “http://valhalla-wireless.com/vwr/.”

[8] AgComm, “http://agcomm.net/.”

[9] S. Asakura, D. Umebara, and M. Kawai, “Distributed location estimation method for mobile terminals using the som algorithm,” *IEICE Journal B*, vol. Vol.J85-B, no. No.7, pp. 1042–1050, July 2002.

[10] P. Bahl and V. Padmanabhan, “RADAR: An inbuilding RF-based user location and tracking system,” in *In Proc. of IEEE INFOCOM, Vol.2*, August 2002, pp. 775–784.

[11] T. Kitasuka, “Wireless LAN basd indoor positioning system WiPS and its simulatioin,” in *In Proc. of 2003 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing - PacRim 2003, Vol.1*, August 2003, pp. 272–275.

[12] D. Niculescu and B. Nath, “Ad hoc positioning system (aps) using aoa,” in *IEEE INFOCOM, Vol.3*, July 2003, pp. 1734–1743.

[13] M. Sakurada, M. Kawahara, and T. Sasaki, “Hybrid ranging method based on combination algorithm of cycle figuration probability and recieved signal strength indication,” *The Transactions of the Institute of Electronics, Information and Communication Engineers. B*, vol. Vol.J95-B, no. No.2, pp. 229–237, February 2012.