Towards Accurate Mobile Sensor Network Localization in Noisy Environments

Harsha Chenji and Radu Stoleru

Abstract—The node localization problem in mobile sensor networks has received significant attention. Recently, particle filters adapted from robotics have produced good localization accuracies in conventional settings. In spite of these successes, state of the art solutions suffer significantly when used in challenging indoor and mobile environments characterized by a high degree of radio signal irregularity. New solutions are needed to address these challenges. We propose a fuzzy logic-based approach for mobile node localization in challenging environments. Localization is formulated as a fuzzy multilateration problem. For sparse networks with few available anchors, we propose a fuzzy grid-prediction scheme. The fuzzy logic-based localization scheme is implemented in a simulator and compared to state of the art solutions. Extensive simulation results demonstrate improvements in the localization accuracy from 20% to 40% when the radio irregularity is high. A hardware implementation running on Epic motes and transported by iRobot mobile hosts confirms simulation results and extends them to the real world.

Index Terms—node localization, wireless sensor networks, mobility, fuzzy logic

1 Introduction

Wireless sensor networks are increasingly a part of the modern landscape. Disciplines as diverse as volcanic eruption prediction [1] and disaster response [2] benefit from the addition of sensing and networking. A common requirement of many wireless sensor network (WSN) systems is localization, where deployed nodes in a network discover their positions.

In some cases, localization is simple. For smaller networks covering small areas, fixed gateway devices and one-hop communications provide enough resolution. Larger networks may be provisioned with location information at the time of deployment [3].

However, in many common environments, localization is more difficult. GPS-based localization may be unreliable indoors, under forest canopies, or in natural and urban canyons. For example, GPS is used for high-precision asset tracking in [4] but fails indoors. Signal strength-based solutions similarly fail when there is a high degree of RF multi-path or interference. [5] relies on accurate measurement of RF TDOA and distance traveled and quickly degrades as accuracy decreases. Radio interferometry localizes nodes to within centimeters in [6] but fails in multipath environments. Mobile beacons roam an outdoor environment in [7] but localization requires a dense network and assumes favorable conditions. All these solutions rely on stable environments with low multipath, where measured or sensed ranges (which are typically obtained by time of arrival, angle of arrival or received signal strength techniques) reliably predict the actual distance between two nodes. For low multipath environments, accurate models have been proposed for estimating time of arrival, angle of arrival and received signal strength [8].

Mobility complicates the localization problem since node to node distance variations *and environment changes* (e.g., due to node mobility or interference from an external source) introduce additional effects, such as small scale fading. Due to the relative motion between mobile nodes, each multipath wave experiences an apparent shift in frequency (i.e., the Doppler shift), directly proportional to the direction of arrival of the received multipath wave, and to the velocity/direction of motion of the mobile [9]. Due to environment changes (i.e., objects in the radio channel are in motion), a time varying Doppler shift is induced on multipath components. Consequently, in such environments affected by small scale fading, it is challenging to use simple connectivity (which itself can vary dramatically [10]) or Received Signal Strength (RSS) for accurate localization.

Fuzzy logic offers an inexpensive and robust way to deal with highly complex and variable models of noisy, uncertain environments. It provides a mechanism to learn about an environment in a way that treats variability consistently. In one well-established fuzzy system, the Sendai railroad [11], fuzzy logic allowed the integration of noisy data related to rail conditions, train weight, and weather into acceleration and braking algorithms. Fuzzy logic can similarly be applied to localization. Empirical measurements are made between participating anchors in predictable encounters. These measurements are analyzed to produce rules that are used by the fuzzy inference systems, which interpret RSS input from

- *The authors are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77845. E-mail: {cjh, stoleru}@cse.tamu.edu*

- *An earlier version of this article was presented at DCOSS 2010.*

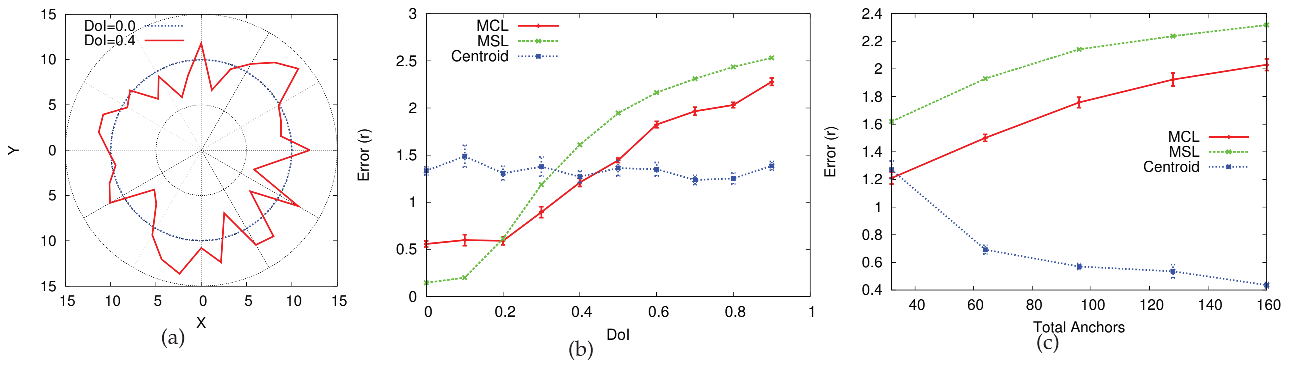

Fig. 1. (a) Illustration of radio patterns for two different degrees of radio irregularity (DoI); (b) the effect of DoI on localization error in MSL, MCL and Centroid; and (c) the effect of anchor density on localization error, at DoI=0.4, for MSL, MCL and Centroid.

unlocalized nodes and other anchors. The output of this process recovers the actual distance, compensated for variability in the local environment. This basic technique is employed in two constituent subsystems of FUZLOC - the Fuzzy Multilateration System (FMS) and the Fuzzy Grid Prediction System (FGPS). The contributions of this article are as follows:

- We formulate the mobile node localization problem for noisy environments, as a fuzzy inference process.
- We present fuzzy multilateration, a component of our fuzzy inference process, which obtains a node's location from noisy RSS measurements, using fuzzy rule sets.
- We present a fuzzy grid prediction scheme, which optimizes our fuzzy inference process, under conditions of low anchor density.
- We demonstrate the feasibility of our proposed technique, through an implementation using mote hardware hosted on iRobot.
- We perform extensive simulations and compare our solution with to state of the art algorithms, using both real-world and synthetic data.

This article is organized as follows. Section 2 motivates our work. In Sections 3 and 4 we present our fuzzy logic-based node localization framework, and the node localization system design, respectively. We evaluate the performance of our proposed node localization system in Section 5, including a real hardware implementation. We review related work in Section 6 and conclude in Section 7. The Supplemental Material contains a detailed explanation of the fuzzification process, a numerical example for fuzzy multilateration, examples of fuzzy rulesets and an experiment showing the effect of Monte Carlo sample size on localization accuracy.

2 Motivation

This article is motivated by our interest in a localization technique for a mobile sensor network, deployed in a harsh environment and a set of interesting/surprising results obtained from simulations of two state of the art localization techniques for mobile sensor networks, namely MCL [12] and MSL [13]. We define a harsh environment as one in which the distance between sender and receiver cannot be accurately determined from the RSS alone, due to environmental phenomena such as multipath propagation and interference.

For more complete *problem formulation* we mention that the aforementioned localization techniques assume that given a set of mobile sensor nodes, a subset of nodes, called anchors, know their location in a 2-dimensional plane. Also, nodes and anchors move randomly in the deployment area. Maximum velocity of a node is bounded but the actual velocity is unknown to nodes or anchors. Nodes do not have any knowledge of the mobility model. Anchors periodically broadcast their locations. All nodes are deployed in a noisy, harsh environment and they do not have any additional sensors except their radios. MCL gathers samples using Monte Carlo methods and filters them using a particle filter, with the criteria being that each sample should be within range of a 1 hop anchor (with respect to itself) while at the same time, not being in range of a 2-hop anchor. Samples are assigned weights over successive iterations. MSL improves upon MCL by using criteria involving all neighbors and not just anchors. MSL is also adaptable to static scenarios if the nodes are allowed to exchange their samples and weights.

Using simulators developed by the authors of [12] [13], we developed a scenario with highly irregular radio ranges, typical of harsh indoor or extremely obstructed outdoor environments. The irregularity in the radio range is modeled in these simulators as a degree of irregularity (DoI) parameter [12]. The DoI represents the maximum radio range variation per unit degree change in direction. An example, depicted in Figure 1(a), when DoI=0.4 the actual communication range is randomly chosen from [0.6r, 1.4r].

Simulation results, for a network of 320 nodes, 32 anchors deployed in a 500×500 grid and moving at 0.2r (r, the radio range) are shown in Figures 1(b) and 1(c). Figure 1(b) demonstrates that the DoI parameter has a significant negative effect on the localization accuracy. At DoI=0, MCL and MSL achieve localization errors of 0.2r and 0.5r. With an increase in the DoI to 0.4, their localization error increases 400%. More surprisingly, as depicted in Figure 1(c), at a high DoI value, an increase in the number of anchors has a detrimental effect on localization accuracy. This result is counter-intuitive since access to more anchors implies that nodes have more opportunities to receive accurate location information, as exemplified by the performance of Centroid (which computes the location as the average of the coordinates of all anchors in its vicinity), in the same figure. A similar observation is made in [7] although no further study was performed. Our results and also those of [14], [15] suggest that large errors are detrimental to the Monte Carlo method since the samples get successively polluted with time. In [15], a proposed Mixture-MCL method uses odometry to gather samples and then uses sensor data to assign weights, enabling it to recover quickly from such errors, while [14] does the same based on error correction based on learnt paths and topological constraints. In the specific case of MCL, the nodes used for filtering the samples may not be actual neighbors because of the non-uniformity in the radio range varies in every direction. The number of polluted samples increases with increasing anchor density. Simply increasing the size of the particle filter in MCL (to 1000 from the current value of 50) does not improve the accuracy significantly, as can be seen in Fig. 10 of [12].

3 A Fuzzy Logic-Based Node Localization Framework

The challenges identified above were partially addressed in recent work in sensor network node localization [16], [17]. The authors create hybrid localization mechanisms that make use of range-based localization primitives (e.g., RSSI) to validate and improve the accuracy of range-free techniques.

In a similar vein, we propose to formulate the localization problem as a fuzzy inference problem by using RSSI to obtain distance, in a fuzzy logic-based localization system where the concept of distance is very loose, such as "High", "Medium" or "Low". The core intuition is that accurate ranges can be determined by learning about the local RF environment and developing rules based on this knowledge. Fuzzy logic provides a simple and computationally inexpensive way to accomplish this learning. In other, similarly dynamic scenarios like rail transportation [11] and photovoltaic power generation [18], fuzzy logic provides mechanisms that allow simple systems to smartly adapt to rapidly changing environments.

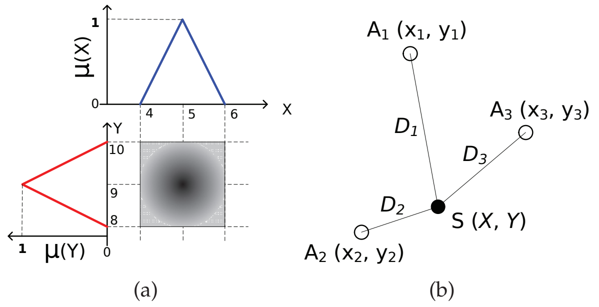

Fig. 2. (a) Representation of a fuzzy location, using two triangular membership functions; and (b) a sensor node S with fuzzy coordinates X and Y, to be located using three anchors at (x_1, y_1), (x_2, y_2) and (x_3, y_3).

In our proposed fuzzy logic-based localization system, distances between a mobile sensor node and anchor nodes are fuzzified, and used, subsequently in a Fuzzy Multilateration procedure to obtain a fuzzy location. In case two or more anchors are not available for performing localization using fuzzy multilateration, the sensor node employs a new technique, called fuzzy grid prediction, to obtain a location, albeit imprecise. In the Fuzzy Grid Prediction method, the node uses ranging information from any available anchor to compute distances to several fictitious "virtual anchors" which are assumed to be located in predetermined grids or quadrants. This allows the node to locate the grid/quadrant in which it is present.

In conventional localization schemes, the location of a node is typically represented by two coordinates that uniquely identify a single point within some two-dimensional area. Localization using fuzzy coordinates follows a similar convention. The two dimensional location of a node is represented as a pair (X, Y), where both X and Y are fuzzy numbers and explained below. However, instead of a single point, the fuzzy location represents an area where the probability of finding the node is highest, as depicted in Figure 2(a). This section develops the theoretical foundation behind the computation of this fuzzy location, using imprecise and noisy RSSI measurements.

3.1 Background

Fuzzy logic revisits classical set theory and modifies it to have non-rigid, or fuzzy, set boundaries. Where classical set theory is concerned with collections of discrete objects, a *fuzzy set*, sometimes called a *fuzzy bin*, is defined by an associated *membership function* μ, which describes the degree of membership $0 \leq \mu(x) \leq 1$ of a *crisp (regular) number* x in the fuzzy set. The process of calculating the membership of a crisp number for many fuzzy sets is called the *fuzzification process*.

A *fuzzy number* is a special fuzzy bin where the membership is 1 at one and only one point. A fuzzy number represents a multi-valued, imprecise quantity unlike a single valued traditional number. One popular $\mu(x)$ function, is the *triangular membership function*:

$$\mu(x) = \begin{cases} 0 & \text{if } x < a \\ (x-a)/(b-a) & \text{if } a \le x \le b \\ (c-x)/(c-b) & \text{if } b \le x \le c \\ 0 & \text{if } x > c \end{cases} \qquad (1)$$

where (a, b, c) defines a triangular bin. In this article, we chose a triangular membership function because, in addition to being a good substitute for the more widely used Gaussian function, it has linear components only and computing membership is less resource intensive, suitable for our resource constrained sensor nodes. Since not all triangular memberships are symmetric, we use the triangular function in its most general form. A comprehensive example can be found in Supplemental Material, Section 1.

A *fuzzy system* translates a crisp input into a fuzzy output using a set of *fuzzy rules* which relate input and output variables in the form of an IF-THEN clause. Typically the IF clause contains the input linguistic variable (e.g., RSSI) and the THEN clause contains the output linguistic variable (e.g., DISTANCE). An example rule is:

IF RSSI is *WEAK* **THEN** DISTANCE is *LARGE*

3.2 Fuzzy Multilateration

As shown in Figure 2(b), consider a node S that wants to be localized, in the vicinity of three anchor nodes A_j $(j = \overline{1,3})$. Each anchor node is equipped with a set of fuzzy rules that map fuzzy RSSI values to fuzzy distance values:

Rule i: **IF** RSSI is $RSSI_i$ **THEN** DIST is $Dist_i$

where $RSSI_i$ and $Dist_i$ are fuzzy linguistic variables (e.g. $WEAK$, $MEDIUM$, $HIGH$).

A fuzzy rule is created when two anchors can communicate directly. Since anchors know their locations, they can find the distance between themselves and also measure the RSSI. The anchors then fuzzify the crisp RSSI and distance values into two fuzzy bins $RSSI_i$ and $Dist_i$ respectively, through the process of fuzzification. The chosen fuzzy bin is the one in which the crisp value will have the highest membership value.

For a more general case, when the node S is within radio range of n anchors, the node localization problem can be formulated as a fuzzy multilateration problem. The following:

$$\begin{aligned} F_1 &= (X - x_1)^2 + (Y - y_1)^2 - D_1^2 = 0 \\ &\cdots \\ F_n &= (X - x_n)^2 + (Y - y_n)^2 - D_n^2 = 0 \end{aligned} \qquad (2)$$

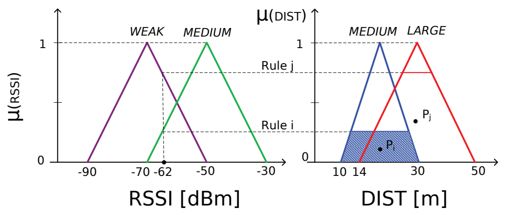

Fig. 3. The fuzzy inference process for an input RSSI value of -62dBm. In this example, the fuzzy rule base maps this value through two rules: "Rule i" and "Rule j". The dotted lines represent fuzzy inference: finding the membership (vertical line on left), applying the same membership to the output bin (horizontal line towards right) and defuzzification (the lines intersect the triangles to form a trapezoid).

defines a non-linear system of equations describing the relation between the locations of the nodes and anchors and the distances among them. The variables X, Y and D_k $(k = \overline{1,n})$ are fuzzy numbers representing the location of the node and the distance to anchors respectively, while (x_k, y_k) $(k = \overline{1,n})$ are crisp numbers representing the crisp location of the anchors. The objective is to minimize the mean square error over all equations.

3.2.1 Fuzzy Inference

A definition of the process of obtaining the fuzzy distance D_k between node and anchor is needed before solving the system of equations. This process, called fuzzy inference, transforms a crisp RSSI value obtained from a packet sent by a node and received by an anchor into a fuzzy number D_k. Figure 3 depicts an example for the fuzzy inference process. As shown, an RSSI value of -62dBm has different membership values $\mu(RSSI)$ for the fuzzy bins $WEAK$ and $MEDIUM$. The two fuzzy bins, in this example, are mapped by a fuzzy rule base formed by two fuzzy rules:

Rule i: **IF** RSSI is $MEDIUM$ **THEN** DIST is $MEDIUM$
Rule j: **IF** RSSI is $WEAK$ **THEN** DIST is $LARGE$

These two fuzzy rules define the mapping from the RSSI fuzzy sets to the DIST fuzzy sets. As shown in Figure 3, the two fuzzy rules indicate the membership $\mu(DIST)$ in the DIST domain. P_i and P_j indicate the center of gravity of the trapezoid formed by the mapping of the RSSI into fuzzy bins $MEDIUM$ and $LARGE$, respectively.

Typically, a single RSSI value triggers multiple fuzzy rules (the membership value of the crisp value in the input bin of the fuzzy rule is non-zero), resulting in multiple distance bins. Assume that the fuzzy rule base maps an RSSI value to a set of m fuzzy Dist bins. The set of centers of gravity P_l $(l = \overline{1,m})$ is denoted by $P = \{P_1, P_2 \ldots P_m\}$. The output fuzzy number D_k is calculated as follows: First, calculate the

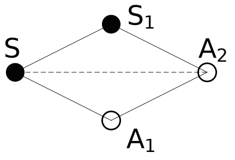

Fig. 4. Illustrating the multi-hop case for fuzzy multilateration: A node S localizes itself using A_2 and A_1.

centroid of all points in P - call it P_c. Next, take the centroid of all points in P whose abscissa is less than that of P_c i.e., $L = \{P_n | x(P_n) \leq x(P_c)\}$. Similarly, $G = \{P_n | x(P_n) \geq x(P_c)\}$ is the set of points whose abscissa is greater than that of P. The abscissae of three points P, L and G represent the resulting fuzzy distance D_k, formally described as (subscript x denotes abscissa):

$$D_k = (a, b, c) = \left(\left(\frac{\sum L_n}{|L|} \right)_x, (P_c)_x, \left(\frac{\sum G_n}{|G|} \right)_x \right) \quad (3)$$

This definition of obtaining a fuzzy number through fuzzy inference produces a fuzzy number while giving more "weight" to the centroid by eliminating some possibilities at the edge. To truly represent the result one would need to compute a smooth and continuous function like the Gaussian membership function, but the triangular approximation has the advantage of reduced computation complexity.

Equation 3 limits its analysis to situations where the anchors and the node desiring localization are one hop from each other. This constraint limits the degree of accuracy that can be achieved. Two hops provide a good trade off between messaging overheads and accuracy as explained later. Consider an anchor A_2 (Figure 4) which is 2 hops away from a node S. Suppose that a regular node S_1 and an anchor A_1 are neighbors of both S and A_2. The aim is now to find the distance $\overline{D_{SA_2}}$. In a 2-dimensional space, a straight line between two points is also the shortest possible; hence a good approximation is the minimum of all known distances between the 2 points. Applying this fact, we can now calculate:

$$\overline{D_{SA_2}} = \min(\overline{D_{SS_1}} + \overline{D_{S_1A_2}}, \overline{D_{SA_1}} + \overline{D_{A_1A_2}}) \quad (4)$$

The distances in Equation 4 are fuzzy values, as the result of defuzzification by either A_1 or A_2 depending on the sender. Addition of two triangular fuzzy numbers (a, b, c) and (d, e, f) is well known in fuzzy logic theory [19] to be the sum of their individual parameters:

$$(a, b, c) + (d, e, f) = (a + d, b + e, c + f)$$

The smallest fuzzy number, to be computed in Equation 4 is simply the fuzzy number with the lowest center value [19]:

$$\min((a, b, c), (d, e, f)) = \begin{cases} (a, b, c) & \text{if } \min(b, e) = b \\ (d, e, f) & \text{if } \min(b, e) = e \end{cases}$$

The minimum of many fuzzy numbers can be recursively computed in the case of multi hop multilateration; it is beyond the scope of this article. In order to solve the non-linear system of Equations 2, in two fuzzy variables, the fuzzy variant of the iterative classical Newton method based on the Jacobian matrix [20] is used. To accomplish this, the fuzzy numbers are expressed in their parametric form $X = (\underline{X}, \overline{X})$ where $\underline{X}$ and $\overline{X}$ are continuous bounded non-decreasing and non-increasing, respectively, functions. These functions effectively represent the "left half" and "right half" of the membership function.

For a triangular membership function, such as defined in Equation 1, a parametric representation in $r \in [0, 1]$ is:

$$X = (a + (b - a)r, c - (c - b)r)$$

The system of Equations 2 is, therefore, represented in the parametric form. Without loss of generality, assume that X and Y are positive. Then, the system can be split into:

$$\begin{aligned} \underline{F_1} &= (\underline{X} - x_1)^2 + (\underline{Y} - y_1)^2 - \underline{D_1}^2 = 0 \\ &\ldots \\ \underline{F_n} &= (\underline{X} - x_n)^2 + (\underline{Y} - y_n)^2 - \underline{D_n}^2 = 0 \end{aligned} \quad (5)$$

and

$$\begin{aligned} \overline{F_1} &= (\overline{X} - x_1)^2 + (\overline{Y} - y_1)^2 - \overline{D_1}^2 = 0 \\ &\ldots \\ \overline{F_n} &= (\overline{X} - x_n)^2 + (\overline{Y} - y_n)^2 - \overline{D_n}^2 = 0 \end{aligned} \quad (6)$$

The Jacobian J is constructed as:

$$J = \begin{bmatrix} \underline{F_1}_{\underline{X}} & \underline{F_1}_{\overline{X}} & \underline{F_1}_{\underline{Y}} & \underline{F_1}_{\overline{Y}} \\ \overline{F_1}_{\underline{X}} & \overline{F_1}_{\overline{X}} & \overline{F_1}_{\underline{Y}} & \overline{F_1}_{\overline{Y}} \\ \ldots & \ldots & \ldots & \ldots \\ \underline{F_n}_{\underline{X}} & \underline{F_n}_{\overline{X}} & \underline{F_n}_{\underline{Y}} & \underline{F_n}_{\overline{Y}} \\ \overline{F_n}_{\underline{X}} & \overline{F_n}_{\overline{X}} & \overline{F_n}_{\underline{Y}} & \overline{F_n}_{\overline{Y}} \end{bmatrix} \quad (7)$$

$$J = \begin{bmatrix} 2(\underline{X} - x_1) & 0 & 2(\underline{Y} - y_1) & 0 \\ 0 & 2(\overline{X} - x_1) & 0 & 2(\overline{Y} - y_1) \\ \ldots & \ldots & \ldots & \ldots \\ 2(\underline{X} - x_n) & 0 & 2(\underline{Y} - y_n) & 0 \\ 0 & 2(\overline{X} - x_n) & 0 & 2(\overline{Y} - y_n) \end{bmatrix}$$

Initial guesses of X and Y can be updated as follows. For every iteration compute a matrix Δ:

$$\Delta = [\underline{h}(r)\ \overline{h}(r)\ \underline{k}(r)\ \overline{k}(r)]^T \quad (8)$$

where $\underline{h}$, $\overline{h}$, $\underline{k}$ and $\overline{k}$ are defined as incremental updates to the initial guess:

$$\begin{aligned} \underline{X}(r) &= \underline{X}(r) + \underline{h}(r) \\ \overline{X}(r) &= \overline{X}(r) + \overline{h}(r) \\ \underline{Y}(r) &= \underline{Y}(r) + \underline{k}(r) \\ \overline{Y}(r) &= \overline{Y}(r) + \overline{k}(r) \end{aligned} \quad (9)$$

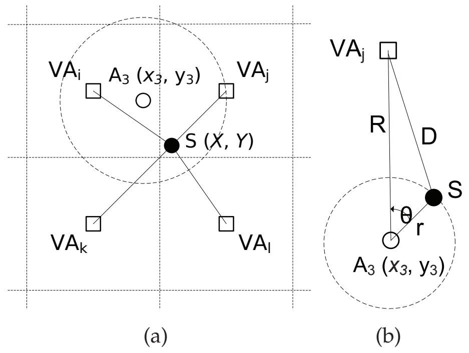

Fig. 5. (a) A sensor S and the grid cells in its vicinity, is within radio range of anchor A_3; and (b) average distance between sensor S and virtual anchor VA$_j$.

The set of equations evaluated at the initial guess is:

$$F = [\underline{F_1}\ \overline{F_1}\ \dots\ \underline{F_n}\ \overline{F_n}]^T \tag{10}$$

The equation that connects them is $\Delta = -J^{-1}F$. The initial guess (X_0, Y_0) is computed from the average of the coordinates of the anchors. Then, J and F are computed for this initial guess. The incremental update Δ is calculated and applied to X and Y. J and F are computed for the new values and the process is repeated until Δ converges to 0 within ϵ.

3.3 Fuzzy Grid Prediction

The multilateration technique presented in the previous section assumes the presence of a sufficient number of anchors, typically three or more. However, in mobile sensor networks with low anchor densities, it might frequently be the case that a node does not have enough anchors for multilateration. To address this problem we extend our fuzzy logic-based localization framework to predict an area, e.g., a cell in a grid, where the node might be. The idea is inspired from cellular systems [21]. We propose to virtualize the anchors, so that a node is within a set of Virtual Anchors at any point in time. A Virtual Anchor is a fictitious anchor which is assumed to located at a known, fixed location in the field of deployment, the distance to which can be found in an approximate way from the node. In FuzLoc, we place virtual anchors at the center of every square cell that the field is divided into, as described below. The key idea is that the nearer a node is to a virtual anchor, the more likely it is that the node can be found in that cell.

Consider the area in which the network is deployed to be subdivided into a grid of G cells, as depicted in Figure 5(a). Denote the probability that a node S is in a cell j $(j = 1 \dots G)$ by p_j. To infer these probabilities, we construct a fuzzy system, whose input is the distance d_j between S and the center of cell j, and the output is a scalar $0 < p_j < 1$ for each j. A rule in our fuzzy system is as follows:

Rule i: **IF** (DIST$_{grd1}$ is D_{i1}) and ... and (DIST$_{grdG}$ is D_{iG}) **THEN** ($PROB_{grd1}$ is P_{i1}) and ... and ($PROB_{grdG}$ is P_{iG})

where D_{ij} is the fuzzy bin representing the distance between the node and the center of cell j, and P_{ij} is the fuzzy bin representing the probability that node S is in cell j.

For each rule i, we calculate p_j by first fuzzifying d_j, applying it to the rule, and then defuzzifying the aggregate, as we described in Section 3.2.1. Once the most probable cell is found, the location of the node can be computed as the intersection between this cell and a circle with a radius of r around the anchor.

It is paramount to remark that we can obtain p_j only if the node S has at least one anchor in its vicinity, i.e., we can estimate D_{ij}. The technique we propose for estimating D_{ij} is described in Section 3.3.1.

Before proceeding with the description of how we compute D_{ij}, we describe how to update p_j when no anchor is in the vicinity of node S. Since there is a high correlation between the current and previous cell a node is in, we construct a Recursive Least Squares (RLS) filter which predicts the cell in which the node S might be. For each cell j, we store a buffer $x_j(k) = [p_j(k)\ p_j(k-1)\ \dots\ p_j(k-m)]^T$ of m previous samples. We then define an RLS filter, updated whenever a new sample $p(k+1)$ is available, as:

$$x_j(k+1) = w_j^T(k)x_j(k)$$

where $w_j(k) = w_j(k-1) + a_j(k)g_j(k)$ is a vector of coefficients, computed as follows:

$$a_j(k) = x_j(k) - w_j^T(k-1)x_j(k)$$
$$g_j(k) = P_j(k-1)x_j(k)\{\lambda + x_j^T(k)P_j(k-1)x_j^T(k)\}^{-1}$$
$$P_j(k) = \lambda^{-1}P_j(k-1) - g_j(k)x_j^T(k)\lambda^{-1}P_j(k-1)$$

where $0 \le \lambda \le 1$ is the forgetfulness factor, a design parameter. $P_j(0)$ is initialized to δI_j, where I is the identity matrix of size $(m+1) \times (m+1)$ and δ is a typically large value.

3.3.1 Calculation of D_{ij}

The fuzzy system requires that we calculate the distance from the node to the virtual anchor. We have to find the average distance instead, because we do not know the node's location. These average distances can be calculated only when at least one anchor is in the node's vicinity.

Consider a node and a sole anchor A_3 which is its neighbor, as illustrated in Figure 5(b). Take the set of all virtual anchors and discard the ones which are at a distance of more than $2R$ from the anchor where R is the radio range of the anchor, since this is the most distant virtual anchor the node can hear in the limiting case where the node is between the anchor and the virtual anchor. To calculate the average distance $\overline{D}$ from the node to a virtual anchor VA_j in

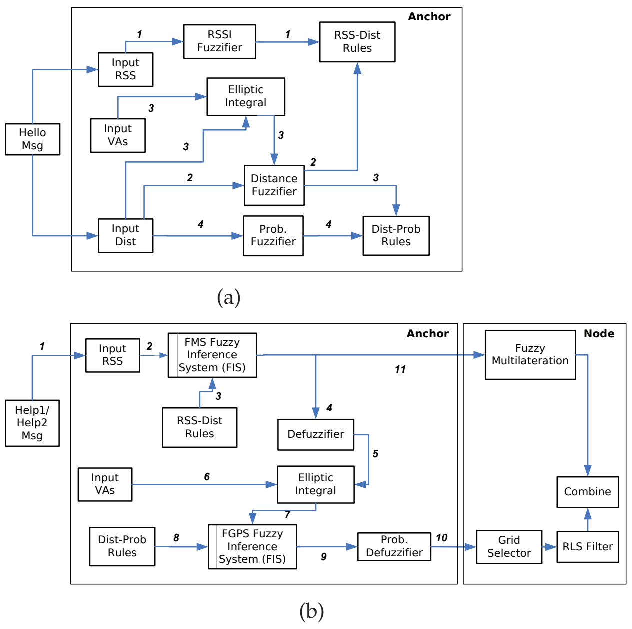

Fig. 6. The fuzzy logic based localization system design with the (a) training; and (b) localization phases.

cell j, we estimate the average distance from VA_j to all points on the circumference of a hypothetical circle around A_3. The radius r of this circle is the defuzzified distance obtained from the fuzzy multilateration system. If the distance between A_3 and VA_j is R, the average distance $\overline{D}$ can be calculated as follows:

$$\begin{aligned}\overline{D} &= \frac{1}{2\pi}\int_0^{2\pi} \sqrt{(R \;-\; r\;\cos\theta)^2 + (r\;\sin\theta)^2}\; d\theta \\ &= \frac{(R-r)}{\pi} E\big[\pi|\frac{-4Rr}{(R-r)^2}\big] \qquad (11)\end{aligned}$$

where $E[x|m]$ is the incomplete elliptic integral of the second kind [22].

This distance $\overline{D}$ to a virtual anchor in cell k for node S is nothing but d_j. When calculated for all j, it becomes the input to the fuzzy system. The output will be a set of probabilities p_j pertaining to each cell. The center of gravity of the lamina defined by the intersection of a circle around the anchor of radius r with the most likely cell is calculated, as explained above. Thus, a location is obtained.

A numerical example for fuzzy multilateration can be found in the Supplemental Material, Section 2.

4 Localization System Design

The node localization system (called FuzLoc) that implements the proposed fuzzy logic-based localization framework is depicted in Figure 6. As shown, the localization system runs on both anchor and sensor nodes. The pseudocode for the localization protocol, as executed by anchors and sensor nodes, is shown in Algorithm 1 and Algorithm 2, respectively.

Figure 6(a) depicts the training phase of Fuzloc while Figure 6(b), the localization phase. Training happens with the participation of anchors only, while the localization phase involves both anchors and nodes. The components required for the fuzzy multilateration subsystem (FMS), as well as the fuzzy grid prediction subsystem (FGPS) are implemented on both anchors and nodes. The fuzzy rules required for these subsystems are created during the training phase. Anchors are assumed to have more computing power than ordinary nodes; they can then maintain these fuzzy rules.

The FuzLoc localization system uses two types of messages - a Hello-type message which anchors *use to train the localization system* (i.e., anchors broadcast their location and build rules), and a Help-type message which nodes *use for localization* (i.e., nodes notify 1-hop and 2-hop anchors and nodes that they need to localize).

The remaining part of this section describes the localization system training (and its use of Hello messages) and the localization protocol execution (and its use of Help-type messages).

4.1 Localization System Training

The training of the localization system takes place every time two anchors come within communication range with each other. The anchors know their locations, hence, an anchor can compute the distance between it and the other anchor. The key observation here is that since an anchor can also measure the RSSI of an incoming message, it can build the fuzzy rules required for both FMS and FGPS. Figure 6(a) depicts the training phase where a single Hello message is used to the build the rulesets for both FMS and FGPS.

FMS - Training: Anchors exchange Hello messages (Algorithm 1, step 2). As shown in Figure 6(a), the RSS of an incoming Hello message ("Input RSS") is fuzzified by choosing the fuzzy set with the highest membership $\mu(RSSI)$ (Figure 6(a), Path 1). The distance between anchors ("Input Dist") is fuzzified into a distance fuzzy set (Figure 6(a), Path 2). The result of the training populates the rule base, i.e., "RSS-Dist Rules" (Figure 6(a), and Algorithm 1, step 7).

FGPS - Training: When an anchor receives a Hello message, it calculates the distances between the sender and each virtual anchor, using Equation 11 (Path 3). This calculation is shown in Algorithm 1, step 9. These distances are then fuzzified ("Distance Fuzzifier", Path 3). Additionally, the probabilities for the anchor being in each grid are updated, as shown in Algorithm 1, step 10. The probabilities are updated based on anchor's real movement, as presented in Section 3.3 (Figure 6(a), Path 4). The computed probabilities are then fuzzified and used for populating (Algorithm 1, step 11) the rules set "Grid-Prob Rules" (Path 4).

Algorithm 1 FuzLoc Protocol - Anchors

```
 1: $[VA]$ ← FGPS.getVirtualAnchors                          ▷ VA of self
 2: BroadcastHello(VA)
 3: procedure RecvHello
 4:     $rss$ ← Radio.getRSS()
 5:     $loc$ ← Message.parseLocation()                      ▷ Loc of sender
 6:     $dist$ ← Distance to sender
 7:     FMS.train(rss, dist)
 8:     $[VA]$ ← Message.parseVA()                           ▷ VA of sender
 9:     $[dist]$ ← Calculate distances to virtual anchors
10:     $[prob]$ ← Calculate probabilities
11:     FGPS.train(dist, prob)
12: end procedure
13: procedure RecvHelp2                                      ▷ Intermediary anchor
14:     Check for cache
15:     $rss$ ← Radio.getRSS()
16:     $[ReplyMsg]$ ← BroadcastHelp1()                      ▷ Rebroadcast
17:     $dist$ ← FMS.getDist(rss)
18:     $[VA]$ ← FGPS.getVirtualAnchors
19:     $[VA.dist]$ ← FPGS.getDists(VA)
20:     $[VA.prob]$ ← FPGS.defuzzify(VA.dist)
21:     Radio.reply(VA.prob,dist,ReplyMsg)
22:     Cache result
23: end procedure
24: procedure ReceiveHelp1(rss1)                             ▷ 2 hop anchor
25:     $rss2$ ← Radio.getRSS()
26:     $dist1$ ← FMS.getDist(rss1)
27:     $dist2$ ← FMS.getDist(rss2)
28:     Radio.reply(dist1,dist2)
29: end procedure
```

Algorithm 2 FuzLoc Protocol - Nodes

```
 1: procedure Localize
 2:     $[info]$ ← BroadcastHelp2()                          ▷ 2 hop HELP
 3:     $anchors$ ← Count(info)
 4:     if $anchors = 0$ then
 5:         $[prob]$ ← Filter.predict()
 6:         $grid$ ← Max(prob).index
 7:         $loc$ ← center(grid)
 8:     else if $anchors = 1$ then                           ▷ FGPS
 9:         $[prob]$ ← info[0].parseVAProb()
10:         TrainFilter(prob)
11:         $grid$ ← Max(prob).index
12:         $dist, center$ ← info[0].parseAnchorLoc()
13:         $circle$ ← ConstructCircle(dist, center)
14:         $loc$ ← SolveIntersection(grid,circle)
15:     else                                                 ▷ FMS
16:         $[dists]$ ← info.parseDistances()
17:         $[centers]$ ← info.parseLocations()
18:         $loc$ ← solveFMS(dists,centers)
19:     end if
20: end procedure
21: procedure ReceiveHelp2                                   ▷ Intermediary Node
22:     Check for cache
23:     $rss$ ← Radio.getRSS()
24:     $[ReplyMsg]$ ← BroadcastHelp1(rss)                   ▷ Rebroadcast
25:     Radio.reply(ReplyMsg)
26:     Cache result
27: end procedure
28: procedure ReceiveHelp1
29:     return                                               ▷ Only for anchors
30: end procedure
```

4.2 Localization Protocol

The localization phase which runs on both anchors and nodes is shown in Figure 6(b). In order to obtain its location, a node sends a HELP2 message (Algorithm 2, step 2). A HELP2 message is meant to trigger actions in nodes/anchors which are 1-hop away. These nodes/anchors perform some calculations (explained below), then rebroadcast a HELP1 message, meant to trigger actions in the 2-hop anchors.

When an anchor receives a HELP2 message (shown as "Help Msg" in Fig. 6(b)), it uses the RSSI of the packet in two ways. First, using Equation 3, the anchor computes the fuzzy distance ("FMS FIS") between itself and the node (Algorithm 1 step 17) (Paths 1, 2, 3, 11). This sequence of steps represents the anchor's implementation of the FMS subsystem. These fuzzy distances, from multiple anchors, are then used by the node to compute its location using the nonlinear system of equations ("Fuzzy Multilateration" box).

Secondly, the anchor defuzzifies the fuzzy distance (from Path 4) into a crisp value by taking the center value of the fuzzy bin ("Defuzzifier", Path 4). This is needed since the elliptic integral method can handle crisp values only. Based on this crisp distance, the anchor calculates the distances between the node sending the Help message and (Section 3.3.1) its virtual anchors (step 19) using the incomplete elliptic integral method (Figure 6(b), Paths 5, 6). This set of crisp distances serves as the input (Section 3.3) to the FGPS FIS (Algorithm 1 step 20) (Figure 6(b), Path 7). The FGPS FIS then computes a vector or grid probabilities using the Dist-Prob ruleset (Figure 6(b), Paths 7, 8) obtained using FGPS training.

This vector of probabilities is then defuzzified to a crisp value (by defuzzifying the individual elements) and compiled into the reply message to be sent back to the node (Paths 10). In the reply to the HELP2 message, as mentioned above, the anchor also includes the fuzzy distance between it and the node (Figure 6(b), Path 11). A vector containing the probabilities for the node being in each of the grids (Algorithm 1 step 21) (Figure 6(b), Path 10), which is essentially the output of FGPS FIS and the reply for the HELP1 message that was broadcast.

The anchor then rebroadcasts a HELP1 message with an empty body (Algorithm 1 step 16). When an anchor receives a HELP1 message, it performs the same steps as before: it defuzzifies the RSSI of the received packet into a distance, and replies with the same (Algorithm 1 steps 25-28). A 2-hop anchor does not invoke its FGPS FIS. In case the HELP1 message contains an RSSI in the body (which happens when the intermediary node is a non-anchor), both the contained RSSI and the packet RSSI are defuzzified and included in the reply.

Once a node receives response(s) to its HELP2 message it decides to compute or predict its location (Algorithm 2 steps 4-18). If the node does not receive a response, it uses the RLS filter to predict the most probable grid it is in (Algorithm 2 steps 5-7). If the node receives a response from one anchor, it computes the center of gravity of the area obtained by intersection between: a) the grid with the maximum probability; and b) the circle with a center at the anchor location and with radius equal to the distance between the anchor and the node (Algorithm 2 steps 9-14). If the node receives two or more responses, it uses fuzzy multilateration to iteratively compute its location (Algorithm 2, steps 16-18).

A node can also be on the receiving end of a HELP2 message - when it is an intermediary node. In this case, the node first detects the RSSI of the received message (Algorithm 2 step 23) and then packages into a HELP1 message and broadcasts it (Algorithm 2 step 24). Any response(s) to this message will be sent back to the sender as a reply to the original HELP2 message that was sent by the node intending to localize. A node ignores any HELP1 message it receives, since it is meant only for anchors (Algorithm 2 step 29).

5 PERFORMANCE EVALUATION

In this section, we first demonstrate that FUZLOC can be implemented and run on real mote hardware, then show FUZLOC's superior performance, when compared with state of art solutions like MCL [12], MSL [13] and Centroid [23]. Owing to the relatively few number of robots, the difficulty in implementing MCL and MSL on real hardware (please note that neither MSL, nor MCL have been implemented/evaluated on real hardware), controlling the anchor and seed density and the physical space constraints, we decided to compare performance of FUZLOC with state of art solutions, in simulations using both empirical and synthetic data.

In the remaining part of this section we present FUZLOC implementation on real-hardware, describe the empirical and synthetic RSSI-Distance mapping, and performance evaluation results.

5.1 System Implementation Validation

We implemented FUZLOC/FMS on EPIC motes running TinyOS 2.1.1. Since the matrices involved in FMS are not always square and hence they cannot be simply inverted, the fast and lightweight SVD based pseudo inverse method [22] was implemented on the motes. Relevant portions of the GNU Scientific Library (GSL) were ported to the MSP430 architecture in order to achieve this goal. The result was a fast method of inverting matrices, providing 4 digits of accuracy when compared to a similar computation on a desktop PC. The 1,574 lines of code fit comfortably in 18,726B.

A Fuzzy Inference System (FIS) consisting of a triangular rule set and center-average defuzzification method was implemented in 19,932B of ROM (including the code required to send and receive messages in the radio) and 1,859B in RAM on EPIC motes running TinyOS 2.1.1. Whenever a packet was received on the onboard radio, the detected RSS was applied to the pre-built ruleset and then defuzzified into a fuzzy distance. The distance and RSSI binset consisted of 8 bins each. The defuzzified distance was equal to that produced by a similar computation on a desktop computer, within rounding errors. The execution time was less than 1 second. This proof of concept implementation of FuzLoc on motes demonstrates its feasibility of implementation on a mote.

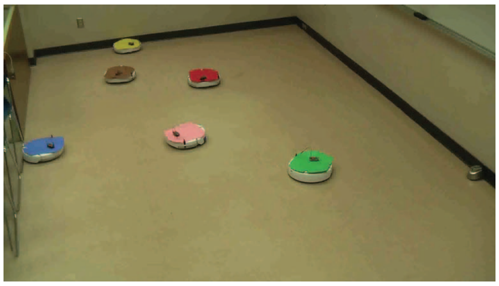
Fig. 7. Experimental setup consisting of 6 iRobot Creates equipped with Epic motes.

5.2 Empirical and Synthetic RSSI-Distance Mapping

For our performance evaluation, we used RSSI-distance mappings obtained from a static sensor network, a small mobile sensor network and from a newly proposed DoI model. They are as follows.

Static Sensor Network. We used a static 42 node indoor testbed, in our lab. RSSI data was collected over 500 iterations with each node beaconing in each iteration. Since the nodes were static, inter-node distances could be calculated easily. This data was used to train and evaluate the FIS, as will be shown in Figure 12(b). However, since only a finite number of unique distances are possible with a static testbed, we decided to use a small mobile testbed as well.

Small Mobile Sensor Network. We collected data (RSSI-distance pairs) using a mobile testbed consisting of 6 “iRobot Creates” and EPIC motes (which interfaced using the serial bus) shown in Figure 7. Over a 125-iteration run, RSSI data was collected between pairs of neighbors at every iteration. In order to get the true locations of the robots (for calculating the distances between them) a digital video camera was used to film the entire experiment in 1080p HD. A small program was written in C and used OpenCV to infer the ground location of the robots using planar homography, since the camera was not in the same plane as the robots.

Each robot has a different color since this makes it easier to track them in the recorded video. Capturing

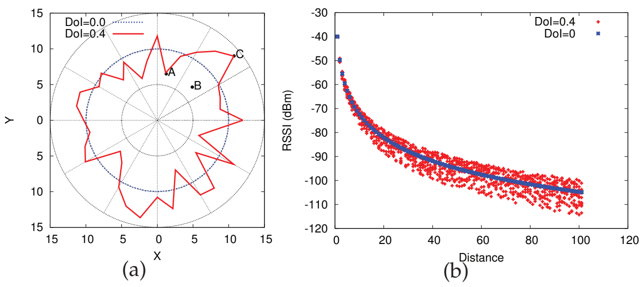

Fig. 8. (a) The DoI model with three points of interest: although A and B are equally distant, their RSS values differ significantly in our EDoI model; and (b) RSSI vs. distance for the radio model used in the simulator, at DoI=0.4 and 0.

the radio effects caused by mobility and the orientation of the antennae on the motes in real time was the main motivation behind the experiment. The ground locations of the robots at each step is then used to infer the actual distance between nodes for every measured RSS between nodes. These RSSI-distance pairs were then used to train and evaluate the FIS as will be shown in Figure 12(c) and described below.

EDoI Model. Since our fuzzy logic-based localization technique makes use of the RSSI, we extended the DoI model [24]. In order to adjust the simulated RSSI for both the actual radio range and log-normal fading, we developed the EDoI model. It combines the general log-normal fading model with the DoI model [24]. In Figure 8(a), $\overline{OA}$ and $\overline{OC}$ are the radio ranges for the antenna situated at the origin O, in two different directions as evaluated by the DoI model. Assume that the receiver sensitivity is -94dBm i.e., if a transmitter with similar characteristics as the receiver is situated at A or C, then the RSSI at the origin will be -94dBm. To calculate the RSSI at a point B in the same direction as C where $\overline{OA} = \overline{OB}$, we apply a log-normal fading model with the reference distance as $\overline{OC}$, such that the RSSI at point C is -60dBm. Note that the RSSI at A is -94dBm, whereas the RSSI at an equidistant point B in a different direction is -60dBm. On top of this, additive random noise (uniformly distributed, min = -20dB, max = 20dB) is applied to the calculated RSSI. This procedure is done every time a node uses this model to simulate an RSSI, ensuring randomness in both temporal and spatial randomness. Formally:

$$RSSI(d) = S_i \frac{\log_{10} d}{\log_{10}[r(1 + DoI \times rand())]} \qquad (12)$$

where S_i is the receiver sensitivity, r is the ideal radio range, DoI is the radio degree of irregularity and $rand$ is a random number $\mathcal{U}[0,1]$.

5.3 Simulation Parameters

Through extensive simulations, we compare our solution with MCL [12], MSL [13] and Centroid [23], since we wanted to evaluate our solution against non-centralized solutions for non-static networks, both Monte Carlo based (MCL, MSL) and simple (Centroid). A theoretical "Perfect FuzLoc" method shows the theoretical optimum FuzLoc can reach, by simply bypassing the FIS and considering the actual distance between nodes. The problem of not having enough anchors in the vicinity of nodes causes non-zero error for Perfect FuzLoc. Data gathered from the the static and small mobile sensor network has been used to evaluate the FIS system. Thus, the FIS system is evaluated using simulated RSSI-Distance data as well as data from the two experiments described before.

We simulate a set (N) of 320 sensor nodes deployed in a 500×500 area. Of the 320 nodes deployed, 32 nodes are designated anchors (set S). The radio range (r) of a node is 50 and the default DoI is 0.4. We chose these simulation parameters for consistency with results reported in [12], [13]. The default receiver sensitivity (S_i) is -94dBm, and a plot depicting the predicted RSSI by our EDoI model, is shown in Figure 8(b). The default maximum node velocity is to 0.2r. This velocity has been reported in [12] to be optimal. We investigate the performance of all solutions for node velocities up to 0.5r. The node velocity is an important parameter since MCL and MSL use it as a filtering criterion in their particle filters. The default setup uses 10 fuzzy triangular bins and the defuzzification method is center-average. The fuzzy location is defuzzified into a crisp location by considering only the center values of the abscissa and the ordinate. The fuzzy bins for distance and RSSI are uniformly distributed between $(0, r)$ and $(-40, -100)$ respectively, with the width of each bin being twice the separation between peaks of two adjacent fuzzy bins. With 10 distance and RSS bins, there are 100 different combinations that can be seen in a RSS-Dist ruleset. Rules encountered more frequently tend to affect the output more than infrequent ones because the defuzzification method involves centroids corresponding to the output bin of each rule. A sample set of fuzzy RSS-Dist rules has been provided in the Supplemental Material, Section 3.

5.4 Radio Irregularity

We performed simulations for different DoI values with all other parameters kept constant. Figure 9 depicts our results, indicating the deterioration in localization accuracy of MCL, MSL and Centroid. The effect of compounded errors due to polluted samples has been investigated as the "kidnapped robot problem" [14] in robot localization. The kidnapped robot test verifies whether the localization algorithm is able to recover from localization failures, as signified by the sudden change in location due to "kidnapping". It has been shown [14] that such uncorrected algorithms collapse when the observed sample is far from

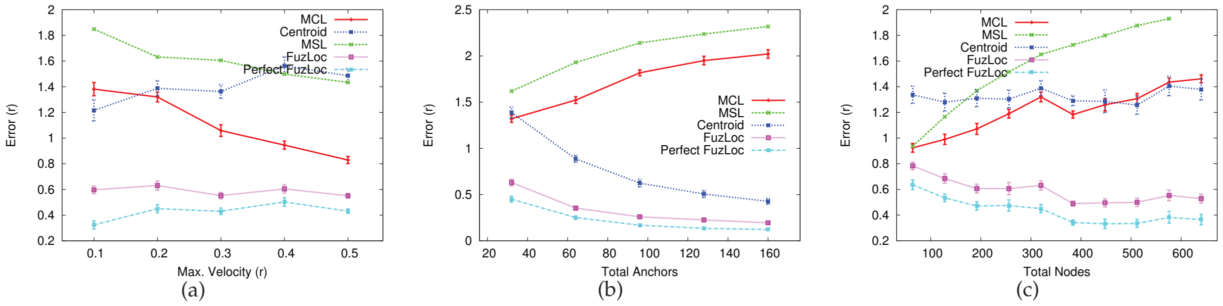

Fig. 10. (a) The effect maximum node velocity has on localization accuracy. (N=320, S=32, DoI=0.4); (b) The effect of anchor density on localization accuracy at DoI=0.4 (N=320, v=0.2r); and (c) The effect of node density on localization accuracy. (S=32, v=0.2r, DoI=0.4)

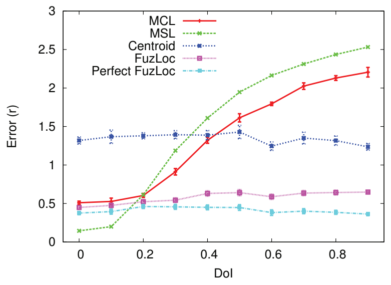

Fig. 9. The effect of DoI on localization accuracy. (N=320, S=32, v=0.2r)

the estimated sample. MSL demonstrates an even more pronounced effect, since it also uses non-anchor neighbors for filtering, thus leading to more pollution. Both FuzLoc and Perfect FuzLoc are unaffected by DoI, with the error of FuzLoc increasing by about 20% at maximum DoI compared to MCL's 300%.

5.5 Maximum Node Velocity

We investigate the effect of maximum node velocity on localization accuracy, for velocities up to 0.5r, a reasonably fast moving speed. The performance results are depicted in Figure 10(a). MCL and MSL assume that nodes know their maximum velocity. Hence, they use the velocity as a filtering condition, which improves their performance. Moreover, high velocity means having more anchors to filter against, leading to the freshening of samples at every instance. Figure 10(a) shows that MCL and MSL decrease their localization error from 1.4r to 0.9r, and 1.9r to 1.4r, respectively. Since Centroid and FuzLoc do not use the velocity, their performance is not expected to improve. Figure 10(a) indicates that their performance is not deteriorating.

5.6 Anchor Density

Anchor density is a critical parameter for anchor-based localization schemes. Figure 10(b) displays the impact of anchor density on the localization schemes where the number of anchors varies from 10% (32 anchors) to 50% (160 anchors), and the DoI is constant at 0.4. The accuracy of MCL and MSL deteriorates because an increase in anchor density is associated with an increase in the number of polluting sources. The mismatch of observed and actual radio ranges causes spurious anchors to appear as node's direct and indirect seeds. MSL considers non-anchor neighbors, hence it experiences higher pollution. Centroid performs better with increasing anchor density, as expected. FuzLoc also has a decrease in localization error, with a larger number of anchors. We observe that FuzLoc is not significantly affected by DoI and ranging errors.

5.7 Node Density

For this performance evaluation scenario we maintained the percentage of anchors fixed at 10%. As shown in Figure 10(c), the evaluated algorithms either suffer or are unaffected. None of the localization algorithms benefits from an increase in the node density. As shown, Centroid and FuzLoc are not substantially affected, except by the inherent randomness in simulation. MCL considers indirect seeds for sampling, hence a high node density means more anchors are misreported as indirect seeds. MSL considers non-anchor neighbors, hence at high node densities, it experiences a huge amount of sample pollution. While non-anchor neighbors help MSL to improve accuracy at low DoI, they become harmful at higher DoI values.

5.8 Number of Bins

The number of bins in the fuzzy system is a design parameter - the greater the number of bins, the higher the accuracy of the system. Our evaluation of the influence of the number of bins is depicted in Figure 11. As shown, as the number of bins increases, the localization error of FuzLoc decreases. This is because more and more RSSs find a bin with high membership. The change in the number of bins, is expected to not affect MCL, MSL, Centroid, or even Perfect FuzLoc. Figure 11 shows that the aforementioned schemes remain invariant whereas FuzLoc

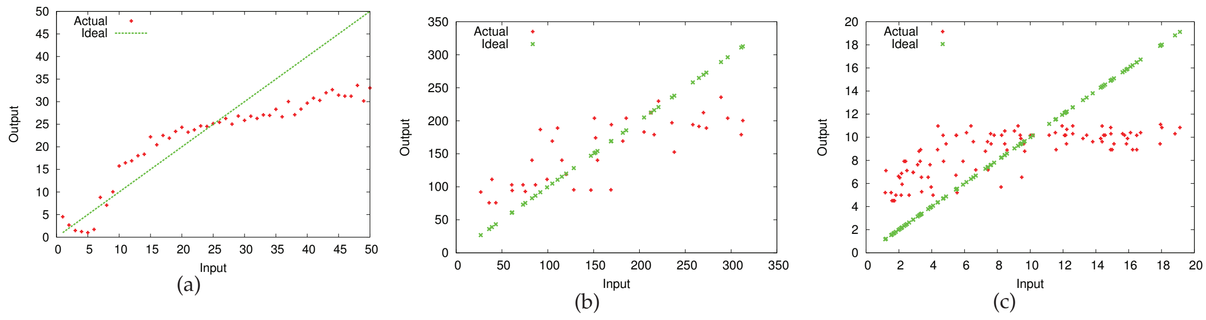

Fig. 12. (a) Performance of the FMS FIS subsystem, with the test input on the X axis and the inferred distances on the Y axis; (b) Performance based on real data gathered from our indoor testbed; (c) Performance based on real data gathered from our mobile testbed consisting of 6 iRobots.

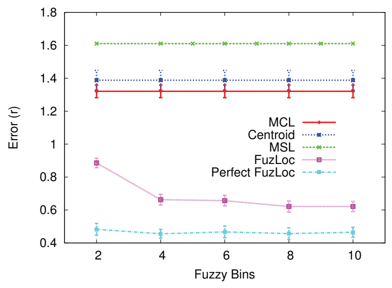

Fig. 11. The effect of the number of fuzzy bins on localization accuracy. (N=320, S=32, v=0.2r, DoI=0.4)

experiences decreasing error with an increase in the number of bins.

5.9 Fuzzy Inference System Performance

Figure 12(a) shows the performance of the FIS engine evaluated using RSSI-distance data generated by the EDoI model, while Figure 12(b) does the same using data from our static testbed and finally, Figure 12(c) uses data from the small mobile testbed. Input distance is on the X axis while the Y axis marks the center value of the defuzzified output distance. After training the system with 30 random RSS-Distance pairs, RSS values deduced from distances were fed into the system so that a distance could be inferred. The straight line shows the ideal case. In order to quantify the accuracy, the root mean squared (RMS) error was calculated and normalized to the radio range. The values are remarkably similar: for the EDoI dataset it was 0.156, for the static network dataset it was 0.182 and for the small mobile testbed it was 0.166. In a way, these numbers reinforce the equivalence of the simulated and real indoor static/mobile radio models, while proving the effectiveness of the EDoI model.

5.10 Overhead

A typical FIS does not require much storage capacity. If there were 8 bins, for example, a single byte could represent a bin. Hence, each FMS rule requires just 2B of storage. Typically, an anchor creates approximately 30 rules during the period of deployment which translates to 60B of storage. The FGPS FIS however, requires 50B for each rule (25 bins in the input, 25 in the output). Note that regular nodes do not store rules, only the anchors store rules. Moreover, due to the nature of the triangular bin shapes, simple calculations are required in order to fuzzify and defuzzify. The only caveat is the inversion of matrices that is required. As for the filter, the node does not construct a filter for all possible cells, since it usually visits a maximum of 4 cells per iteration. Hence, the storage required by a 3-order filter on each non-anchor node will be $(288\times4\times4\times3)$ = 13,824B. MCL requires at least 50 samples for low localization error. Each sample requires a weight. Centroid does not store any history and thus has the smallest storage requirement. Amorphous stores announcements made by the anchors which are flooded throughout the network. If there are 320 nodes, 32 of which are anchors, MCL requires each node to store 50 samples. Each sample has an abscissa and an ordinate, each of at least 4B. Hence, MCL requires around $(50\times4\times2\times320)$ = 128,000B. Fuzzy on the other hand requires around 1,500B for FGPS and around 60 for FMS, with 13,824B for the filters, which sums up to $(1,560\times32 + 13,824)$ = 63,744B which is roughly 50% of the storage MCL requires, and even less than what MSL requires, since MSL mains closeness values.

The communication overhead for 2-hop anchor discovery is the same as that of MCL, and less that of MSL (since MSL needs to exchange samples in addition to anchor discovery). When FuzLoc uses only 1-hop anchors, the communication overhead required is significantly lower since all that is needed is a simple broadcast. Still, FuzLoc performs better than MCL as can be seen in [25]. Therefore, systems desiring lesser communication overhead should use anchors within 1 hop only, while those desiring higher accuracy need to consider anchors within 2 hops. In no state will the communication overhead required by FuzLoc exceed that of MCL or MSL.

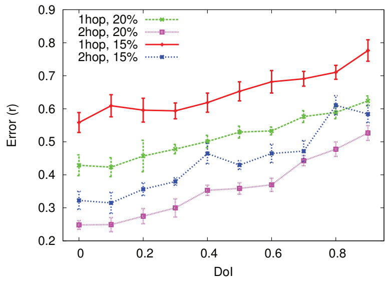

Fig. 13. Comparison of 1 and 2 hop FuzLoc variants at seed densities of 15% (S=48) and 20% (S=64) in a 320 node (N) network across multiple DoIs.

5.11 Single hop and Dual hop FMS

Figure 13 compares the 1-hop and 2-hop variants of FuzLoc. Being a multilateration based method, the presence of a sufficient number of anchors in a node's vicinity is crucial to reducing the error in location estimation. A simple way to ensure this is to increase the percentage of anchors in the network. However, the addition of anchors may be cost prohibitive. A simpler way and less costly solution is to consider anchors which are two hops away. The additional cost incurred for this solution is higher messaging overhead. Instead of traveling over a single hop, localization request broadcasts must take two hops to reach the outer anchors. Replies are consolidated, so no additional messaging is incurred in the reply phase. The number of additional transmissions required vary based on node and anchor density. Figure 13 shows that merely considering the 2-hop anchors results in a much lower error due to the increased number of anchors, than introducing more anchors, across all DoIs. Note that although the number of messages increases, the error is more than halved.

6 Related Work

Range-based localization methods require an estimate of the distance or angle between two nodes to localize and may operate in both absolute and relative coordinate systems. Methods requiring specialized hardware include precise measurement of acoustic phase difference [26], optical sensors/reflectors [27]. Typical drawbacks for these methods include higher computational loads, increased node size, higher energy consumption and increased cost. A lighter weight solution uses fuzzy logic to locate cellular phones in a hexagonal grid in a cellular network [28]. It assumes a fixed number of anchors but handles mobility very well. The computation and refining are not suitable for a resource-constrained computation platform like a MicaZ node. This was the inspiration for this work.

[29] uses precise infra-red ranging in combination with a grid based fuzzy logic approach. [30] proposes using RSS-Distance fuzzy rules to perform crisp localization, when there are anchors placed at the four corners of the deployment area. In [31], time-of-arrival and RSS data is fused together using Bayesian inference, following which fuzzy optimization is used to compute a crisp location. Compared to the above three works, FuzLoc does not assume the presence of additional sensing capabilities [29] [31] and provides localization solutions when there are less than three anchors [30] as well as computing the location as an area, a feature not found in previous work.

Range-free localization methods are typically used in systems where connectivity is the metric of choice and actual geographic distance is less important. Hop counting is a technique frequently used in these scenarios, where the distance between two nodes is inferred from the number of hops a packet takes and is based on some assumed or measured average hop length [32]. A major drawback is that it fails in networks with irregular topologies such as those with a concave shape [33]. Mobility incurs large overhead since all hop counts must be refreshed frequently.

[34] uses hypothesis testing to infer the location of a node, by using RSS PDF distributions gathered by anchors nodes through surveying. The fundamental difference between fuzzy logic and hypothesis testing is that while the former gathers learned intelligence and applies it to a given input, the latter tests all possibilities using tools like the Generalized Likelihood Ratio Test (GLRT). In [34], building each PDF by surveying requires a large sample space. The computation involved in building the PDFs and performing the tests is not suitable for embedded devices unlike FuzLoc. The complexity associated with hypothesis testing increases with increasing number of anchors as well as the deployment area, since there are more tests to be conducted (with multiple tests, the accepted hypothesis from the previous test is used in the next test). With fuzzy logic, the number of rules increases with the anchor density but is independent of deployment area. Fuzzy rules require only storage while hypothesis testing requires computation and storage.

This paper extends previous work done on FuzLoc [25]. Major changes include system implementation on Epic motes, FIS evaluation using mobile iRobots, a distributed protocol, system integration and most importantly, evaluation of and support for fuzzy multilateration using multi hop anchors.

7 Conclusions

We have proposed FuzLoc, a fuzzy logic based localization method suitable for wireless sensor nodes that are mobile in noisy, harsh environments. The constituent systems use fuzzy multilateration and a grid predictor to compute the location of a node as an area. The RSS is cast into bins which encode the imprecision; these bins are subsequently used in our mathematical framework. We remark here that the case of static anchors, considered by neither MCL, nor MSL, will be investigated in future work.

Our method has been evaluated based on a variety of metrics. They prove that our method is resistant to high DoI environments while providing a low localization error without any extra hardware. Only anchors need to have a slightly higher storage requirement. A deployment with more anchors at high DoI decreases the error. The ability to localize using both single-hop and two-hop anchors greatly increases the variety of topologies where localization succeeds. The system implementation proves that the algorithm functions well on resource constrained devices.

Acknowledgements: The authors thank D. Evans and M. Rudafshani for simulator code, and S. George for help and feedback during manuscript preparation. This work was supported, in part, by NSF grants grants 0923203, 1127449.

REFERENCES

[1] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *Internet Computing, IEEE*, March-April 2006.

[2] S. George, W. Zhou, H. Chenji, M. Won, Y. O. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, "Distressnet: a wireless ad hoc and sensor network architecture for situation management in disaster response," *Communications Magazine, IEEE*, March 2010.

[3] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. Stankovic, T. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: an integrated sensor network system for energy-efficient surveillance," *ACM Trans. Sens. Netw.*, 2006.

[4] D. Balakrishnan, A. Nayak, P. Dhar, and S. Kaul, "Efficient geo-tracking and adaptive routing of mobile assets," in *HPCC*, 2009.

[5] H. Akcan, V. Kriakov, H. Brönnimann, and A. Delis, "GPS-free node localization in mobile wireless sensor networks," in *MobiDE*, 2006.

[6] B. Kusy, J. Sallai, G. Balogh, A. Ledeczi, V. Protopopescu, J. Tolliver, F. DeNap, and M. Parang, "Radio interferometric tracking of mobile wireless nodes," in *MobiSys*, 2007.

[7] A. Baggio and K. Langendoen, "Monte carlo localization for mobile wireless sensor networks," *Ad Hoc Netw.*, 2008.

[8] D. Puccinelli and M. Haenggi, "Multipath fading in wireless sensor networks: measurements and interpretation," in *IWCMC*, 2006.

[9] T. S. Rappaport, *Wireless Communications: Principles and Practice (2nd Edition)*, 2nd ed. Prentice Hall, Jan. 2002.

[10] K. Whitehouse, C. Karlof, and D. E. Culler, "A practical evaluation of radio signal strength for ranging-based localization," *Mobile Computing and Communications Review*, 2007.

[11] H. Oshima, S. Yasunobu, and S.-i. Sekino, "Automatic train operation system based on predictive fuzzy control," in *IEEE AI for Industrial Applications*, 1988.

[12] L. Hu and D. Evans, "Localization for mobile sensor networks," in *MobiCom*, 2004.

[13] M. Rudafshani and S. Datta, "Localization in wireless sensor networks," in *IPSN*, 2007.

[14] S. Engelson and D. McDermott, "Error correction in mobile robotmap learning," in *ICRA*, 1992.

[15] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[16] K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasan, "Ecolocation: a sequence based technique for rf localization in wireless sensor networks," in *IPSN*, 2005.

[17] Z. Zhong and T. He, "Achieving range-free localization beyond connectivity," in *SenSys*, 2009.

[18] J. Li and H. Wang, "Maximum power point tracking of photovoltaic generation based on the fuzzy control method," in *SUPERGEN*, 2009.

[19] A. Kaufman, M. Gupta, and B. Esposito, *Introduction to Fuzzy Arithmetic: Theory and Applications*. Van Nostrand Reinhold Company, 1991.

[20] J. Shokri, "On systems of fuzzy nonlinear equations." *Appl. Math. Sci. (Ruse)*, 2008.

[21] X. Shen, J. W. Mark, and J. Ye, "User mobility profile prediction: an adaptive fuzzy inference approach," *Wirel. Netw.*, 2000.

[22] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1964.

[23] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, 2000.

[24] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *MobiCom*, 2003.

[25] H. Chenji and R. Stoleru, "Mobile sensor network localization in harsh environments," in *DCOSS*, 2010.

[26] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, Ákos Lédeczi, G. Balogh, and K. Molnár, "Radio interferometric geolocation," in *SenSys*, 2005.

[27] R. Stoleru, P. Vicaire, T. He, and J. A. Stankovic, "Stardust: a flexible architecture for passive localization in wireless sensor networks," in *SenSys*, 2006.

[28] X. Shen, J. W. Mark, and J. Ye, "Mobile location estimation in cdma cellular networks by using fuzzy logic," *Wirel. Pers. Commun.*, 2002.

[29] A. Dharne, J. Lee, and S. Jayasuriya, "Using fuzzy logic for localization in mobile sensor networks: simulations and experiments," in *ACC*, 2006.

[30] S.-Y. Chiang and J.-L. Wang, "Localization in wireless sensor networks by fuzzy logic system," in *KES*, 2009.

[31] S.-L. Dong, J.-M. Wei, T. Xing, and H.-T. Liu, "Constraint-based fuzzy optimization data fusion for sensor network localization," in *SKG*, 2006.

[32] D. Niculescu and B. Nath, "DV Based Positioning in Ad Hoc Networks," *Telecommunication Systems*, January 2003.

[33] C. Wang and L. Xiao, "Sensor localization in concave environments," *ACM Trans. Sen. Netw.*, 2008.

[34] I. C. Paschalidis, K. Li, and D. Guo, "Model-free probabilistic localization of wireless sensor network nodes in indoor environments," in *MELT*, 2009.

Harsha Chenji Harsha Chenji joined the Department of Computer Science and Engineering at Texas A&M University in August 2007. He is currently a Ph.D. candidate under the guidance of Dr. Radu Stoleru, after graduating with a M.S. (Computer Engineering) degree in Dec 2009. He obtained his Bachelor of Technology in Electrical and Electronics Engineering from the National Institute of Technology Karnataka, Surathkal, India in May 2007.

Dr. Radu Stoleru Dr. Radu Stoleru is currently an assistant professor in the Department of Computer Science and Engineering at Texas A&M University. He received his Ph.D. in computer science from the University of Virginia in 2007, and the Computer Science Outstanding Graduate Student Research Award in 2007. Dr. Stoleru's research interests are in deeply embedded wireless sensor systems, distributed systems, embedded computing, and computer networking. He has authored or co-authored over 60 conference and journal papers with over 1,700 citations.

Supplemental Materials

Towards Accurate Mobile Sensor Network Localization in Noisy Environments

Harsha Chenji and Radu Stoleru

1 Background on Fuzzification

The process of calculating the membership of a crisp number for many fuzzy sets is called the *fuzzification process*.

A *fuzzy number* is a special fuzzy bin where the membership is 1 at one and only one point. A fuzzy number represents a multi-valued, imprecise quantity unlike a single valued traditional number. One popular $\mu(x)$ function, is the *triangular membership function*:

$$\mu(x) = \begin{cases} 0 & \text{if } x < a \\ (x-a)/(b-a) & \text{if } a \le x \le b \\ (c-x)/(c-b) & \text{if } b \le x \le c \\ 0 & \text{if } x > c \end{cases} \qquad (1)$$

where (a, b, c) defines a triangular bin. As shown in Figure 1, the $WEAK$ fuzzy set can be represented as (-90, -70, -50) and $MEDIUM$ as (-70, -50, -30). A crisp number, RSSI = -55dBm has a membership of 0.25 in $WEAK$ and 0.75 in $MEDIUM$.

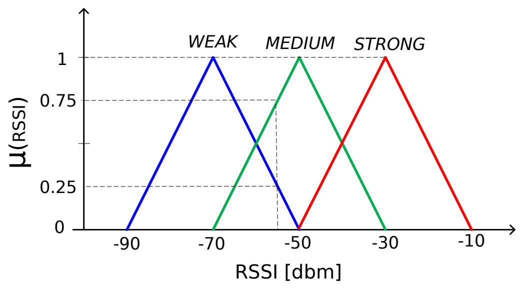

Fig. 1. Fuzzification of a crisp value of -55dBm into fuzzy bins $WEAK$, $MEDIUM$ and $STRONG$ having triangular membership functions: $WEAK$, $MEDIUM$ and $STRONG$ with degrees of membership of 0.25, 0.75 and 0.0 respectively.

2 Fuzzy Multilateration Numerical Example

A numerical example elaborates the concept of fuzzy multilateration. Consider three anchors forming an isosceles triangle (Figure 2) - $(0, 0)$, $(10, 0)$ and $(5, 15)$. Let the node to be localized be at the centroid which is $(5, 5)$. The actual distances to the anchors would be $5\sqrt{2}, 5\sqrt{2}, 10$ respectively. Assume that because of defuzzification, the node calculates the fuzzy distances as $(6, 7, 8), (6, 7, 8), (9, 10, 11)$ respectively. The the system of equations can be expressed as:

$$\begin{aligned} F_1 &= (X-0)^2 + (Y-0)^2 - (6,7,8)^2 = 0 \\ F_2 &= (X-10)^2 + (Y-0)^2 - (6,7,8)^2 = 0 \\ F_3 &= (X-5)^2 + (Y-15)^2 - (9,10,11)^2 = 0 \end{aligned}$$

The actual fuzzy location (X, Y) of the node is $(5, 5, 5), (5, 5, 5)$. Assume the initial guess of (X, Y) to be $(5, 6, 7), (5, 6, 7)$ for the purposes of demonstration, which by the process of convergence should ideally yield the actual location. The parametric form would then be $(5 + r, 7 - r)$. Expanding only F_1 for brevity,

$$\begin{aligned} \underline{F_1} &= (\underline{X} - 0)^2 + (\underline{Y} - 0)^2 - \underline{(6,7,8)}^2 \\ \overline{F_1} &= (\overline{X} - 0)^2 + (\overline{Y} - 15)^2 - \overline{(6,7,8)}^2 \end{aligned}$$

which can then be simplified to

$$\begin{aligned} \underline{F_1} &= (5 + r - 0)^2 + (5 + r - 0)^2 - (6 + r)^2 \\ \overline{F_1} &= (7 - r - 0)^2 + (7 - r - 15)^2 - (8 - r)^2 \end{aligned}$$

Similarly the Jacobian can now be constructed as (first 2 rows only):

$$J = \begin{bmatrix} 2(5+r) & 0 & 2(5+r) & 0 \\ 0 & 2(7-r) & 0 & 2(7-r) \end{bmatrix}$$

The pseudo-inverse of a matrix with symbolic elements is computationally expensive, especially for embedded sensor nodes. Instead of inverting J which contains a symbolic element r, two non-symbolic inverses can be computed (for $r = 0$ and $r = 1$) and the results combined. This alternate computation incurs a loss of accuracy because the solution will be a perfect triangular fuzzy number and not a fuzzy

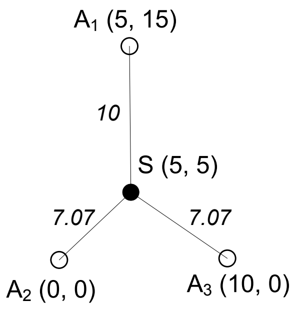

Fig. 2. Three anchors and a node used in the numerical example.

number with little variation. However, the accuracy lost is extremely small.

A simple substitution of $r = 0$ and $r = 1$ in J and F yields:

$$\Delta_0 = J_0^{-1}F_0 \text{ and } \Delta_1 = J_1^{-1}F_1$$

Given a solution (X, Y) expressed in simple form as (x_A, x_B, x_C) and (y_A, y_B, y_C), then:

$$\Delta_0 = \begin{bmatrix} \delta x_A \\ \delta x_C \\ \delta y_A \\ \delta y_C \end{bmatrix} \quad \text{and} \quad \Delta_1 = \begin{bmatrix} \delta x_B \\ \delta x_B \\ \delta y_B \\ \delta y_B \end{bmatrix} \tag{3}$$

where δx_A is the incremental update to x_A. This is obvious since the left half of any fuzzy number in parametric form $(a + (b - a)r)$ evaluates to a when $r = 0$. The same argument holds for the right half. After this step, the new (x_A, x_B, x_C) and (y_A, y_B, y_C) are the input for the next iteration. The process is repeated until sufficient accuracy is obtained. Upon running our algorithm for 10 iterations with the above initial guess, the final location of the node was calculated as $(5.0, 5.0, 5.0), (4.72, 4.97, 5.14)$. The value that would have been used for comparison with algorithms that do not compute an area as a node's location will be the center values of the two fuzzy numbers, i.e, $(5, 4.97)$.

3 Fuzzy Rulesets Example

Table 1 shows a summary of the fuzzy RSS-Dist rules encountered during simulation, on a randomly chosen anchor. The DoI at the time of rule capture was 0.4. The bin indexes refer to the values in an increasing order. The RSSI bin with the lowest index captures the lowest RSS values (-100dBm) while the distance bin counterpart captures the smallest distance (0-0.1R).

We see that rules with a high distance bin index typically have a low RSSI bin index and vice versa. This is representative of the fact the higher the distance, lower is the RSSI and vice versa. The first ten rules stored on this node are shown below:

		Fuzzy Dist Bin Index									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Fuzzy RSSI Bin Index	**0**	0	0	0	0	0	4	6	4	0	0
	1	0	0	0	5	11	14	13	7	1	0
	2	0	0	3	21	10	3	0	0	0	0
	3	0	3	7	1	0	0	0	0	0	0
	4	0	2	0	0	0	0	0	0	0	0
	5	0	1	0	0	0	0	0	0	0	0
	6	1	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0
	8	1	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0

TABLE 1
Distribution of fuzzy RSS-Dist rules, with the RSS bin index (vertical) and the Dist bin index (horizontal), for a total of 117 rules.

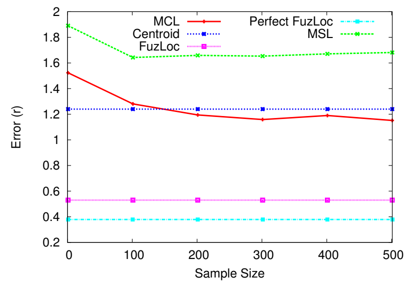

Fig. 3. The effect of the number of samples on localization accuracy. (N=320, S=32, v=0.2r, DoI=0.4)

Rule 0 : If RSS is $RSSI_0$, then DIST is $Dist_7$
Rule 1 : If RSS is $RSSI_0$, then DIST is $Dist_6$
Rule 2 : If RSS is $RSSI_0$, then DIST is $Dist_5$
Rule 3 : If RSS is $RSSI_0$, then DIST is $Dist_6$
Rule 4 : If RSS is $RSSI_0$, then DIST is $Dist_6$
Rule 5 : If RSS is $RSSI_0$, then DIST is $Dist_6$
Rule 6 : If RSS is $RSSI_0$, then DIST is $Dist_5$
Rule 7 : If RSS is $RSSI_0$, then DIST is $Dist_7$
Rule 8 : If RSS is $RSSI_0$, then DIST is $Dist_7$
Rule 9 : If RSS is $RSSI_0$, then DIST is $Dist_6$

4 Effect of Sample Size on Localization Accuracy

Increasing the sample size for Monte Carlo based localization algorithms typically reduces the localization error. However, there is a storage requirement for each node in storing the samples. The effect of sample size on MCL and MSL is shown in Figure 3. As the sample size increases from 1 to around 150, localization error decreases for MCL but fails to decrease noticeably once the sample size is increased further. A similar trend is observed for MSL with the exception that the cutoff size is around 100 samples. The default sample size for both MCL and MSL is 50, in keeping with the choice of the respective authors. At 400 samples, FuzLoc's error is less than half of both MCL and MSL.

Storage overhead for MCL increases linearly with the number of samples. As discussed in Section 5.10 of the main article, MCL requires 128,000B of storage when each node stores 50 samples, compared to 63,744B for FuzLoc. When the sample size is increased to 100, the storage requirement is doubled (100% increase) whereas the error decreases by only 3%. For 400 samples, a 700% increase in storage requirement facilitates a 9.8% decrease in error (baseline of 50 samples). This experiment therefore demonstrates that an increase in sample size does not justify the improvement in accuracy when storage overhead is considered.