Node Localization for Indoor Tracking using Artificial Neural Network

Riya Samanta[*], Chandni Kumari[†], Novarun Deb[‡], Sagar Bose[§],Agostino Cortesi[¶] and Nabendu Chaki[‖]
[*†‖]University of Calcutta, Kolkata, India
Email: [*]study.riya1792@gmail.com, [†]cu.chandni@gmail.com, [‖]nchaki@gmail.com
[‡¶]Ca' Foscari University, Venice, Italy
Email: [‡]novarun.db@gmail.com, [¶]cortesi@unive.it
[§]Homet Solutions, Kolkata, India
Email: [§]sgrbose@gmail.com

***Abstract*—Wireless sensor network (WSN) always comes up with the need of deploying either mobile or immobile sensor nodes or both. Wireless communication among these nodes is crucial and it requires identifying the location of these nodes within a specific region. Global positioning system (GPS) is widely used for location tracking. However, when it comes to WSN, GPS has its limitations, due to its high power consumption and the overhead of additional hardware cost. The research challenge here lies in the efficient location tracking of wireless sensor nodes, especially in closed indoor and outdoor environments. This paper comes up with a simple and easy-to-implement technique using artificial neural networks (ANNs) to manipulate the location of the sensor nodes. In this paper, the back-propagation network training algorithm for providing supervised learning to multilayer perceptron is generalized to synthesize the WSN and gives out 2D Cartesian coordinates of the nodes. The technique is both cost-efficient and achieves 98% accuracy.**

***Index Terms*—Node Localization, ANN, Range-based, Range-free, GPS, Back propagation, Multilayered perceptron, Indoor tracking, Wireless Sensor Network.**

NOTATION

I_m	net Input
W_{km}	weight between k^{th} and m^{th} neuron
O_k	output of k^{th} neuron
θ_m	bias of m^{th} neuron
l	learning rate
Err	error at certain layer instance
AN_i	i^{th} anchor node

I. INTRODUCTION

Node localization is a common issue which has been opted to be solved by various already suggested techniques. Such techniques are broadly grouped into range-free techniques and range-based techniques. Even usage of GPS is also popular in this respect [1],[2],[3]. It is known that range-based techniques are generally based on some physical range metrics like signal strength or angular orientation or distance gap. Some examples of range-based techniques include time of arrival (TOA) method, time difference of arrival (TDOA) method, the received signal strength indicator (RSSI) method and angle of arrival (AOA) method [4]. On the other hand, range-free techniques are based on abstract parameters like hop count and other such hop based information and do not rely on physical range metrics [5]. Range based techniques are very accurate but expensive whereas range-free techniques are less costly but are not that much accurate [1],[2]. Nowadays, smart appliances are equipped with GPS modules, but when it comes to handling sensor applications, assigned with stand alone role, incorporating them with GPS would not be a good idea. This is because GPS works on the principle of line of sight for which the resultant output is processed in the form of latitudinal and longitudinal values. Moreover, GPS signals may not always penetrate thick concrete walls or densely covered forest vegetation and this may fail to provide sufficient coverage [6]. However, it can give results for most outdoor environments, but there is a need for deploying expensive hardware modules or accessories and involvement of huge consumption of power. Thus, usage of GPS on all nodes for large networks consisting of very small, cheap, low power dependent devices should be reduced as far as possible [3]. In this paper, we choose the ANN model to replace the above mentioned methods and derive a technique for location tracking that is both cost efficient as well as accurate.

In [3], the authors demonstrate an experiment on a simple radio based model that exploits non-GPS based methodologies. The results do not correlate well with the reality. WLANs have been deployed for indoor tracking purposes as well [6]. Besides, range-free and range-based techniques are very common to work out node localization [1]. There is a range-free localization algorithm whose average location estimation error (LEE) achieves a LEE average and variance of about zero when the number of sensors is large enough, thereby achieving a very accurate performance among other range-free techniques [5]. There has already been a significant amount of research to solve the location problem in WSNs. A back-propagation learning algorithm (BPLA) has been presented in [7] that exploits a negative gradient descent method making the objective function recognize the parameters in the steepest descent. Moreover, experiments have also been deployed that implement popular training algorithms namely Levenberg-Marquardt and Resilient Backpropagation to achieve the same goal [8]. The use of Euclidean distance as a metric to measure the distance between nodes is proved to be a standard and simple approach [4]. A detailed comparison between various well known algorithms used for node localization has already been chalked out in [9]. Recently, Eugenio and Cortesi [10] analyzed energy consumption of mobile devices when walking in a WiFi network area to study the dynamic of power absorption during exchange of data.

Node localization can be performed in two strategies: *Fine Localization*, which is dependent on approximation of the distance or angular measure between neighboring nodes (provided as input) and *Approximate Localization*, which is free from any input measures as the connectivity between the nodes is used as indicator to approximate the node coordinates [9]. We can identify the following classification of parameters for localization techniques:

- Mobility of nodes (static or dynamic).
- Architecture of deployment of the nodes (centralized or distributed), and
- Technology of measures (range-based or range-free).

II. Scope of the work

As already discussed, the use of GPS in generalized sensor networks is not desirable particularly for handling indoor tracking issues. This is because when GPS is used to track two different nodes within the same floor, both the nodes would be visible at the same point on the map. Here comes the need for some 2D mapping technique using Cartesian coordinates, for plotting two distinct nodes at two different locations. In this paper, we propose an ANN based model for tracking the location of sensor nodes in indoor environments with high accuracy.

III. Proposed model

It is known that ANN has an analogy with the human neural system. Thus, it is capable of learning and can adapt [7],[11] and this property of ANN has been exploited in this work. There have been numerous models of ANN based on the connectivity of the neurons [11],[12],[13]. Back-propagation network learning algorithm provides supervised training to multilayer perceptron. It is simple to design and an efficient error handling strategy. The algorithm has been generalized and developed for WSN. In our experimental settings, the network consists of two kinds of senor nodes:

1) `Anchor nodes` which are aware of their locations and can be implemented by attaching GPS modules with them, and
2) `Generic sensor nodes` which are position-unaware and are dependent on the anchors for obtaining their locations.

The anchors are assigned as the 2D Cartesian coordinate points based on a specific 100 by 100 grid area for our simplicity. Training is based on the minimization of errors between the output generated by the network and the target output. This is achieved by propagating the errors backward, calculated in the previous iterations through the network. Sensor nodes are assumed to be placed at the intersection points of the grid. Euclidean distance formula (1) is used for calculating the distance between the `anchor` and the `sensor` nodes. This is the training dataset, which is the input dataset along with the target dataset representing the actual locations of the nodes.

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

The efficiency of this approach is quite up to the mark on similar research work [7].

IV. Training of the network

The ANN is trained to locate each sensor node in the grid by estimating the distance relative to each defined anchor node and then tested. MATLAB tool is used for the programming purpose. Firstly, log-sigmoid is used as transfer functions in the hidden layer with 10 to 15 neurons. It is known that log-sigmoid transfer function, specifically "logistic" function is used in ANN models to incorporate non-linearity, so that the output can be squashed onto the range $[0, 1]$. Moreover, logistic transfer function satisfies the property between its derivative and itself which makes it easier for computation, particularly for learning algorithms. The range $[0, 1]$ is taken into consideration for processing as manipulation of the result becomes easy and chances of convergence become high. Secondly, linear function is used for the output layer so as to allow the flow of processed values to the next level without much non-linearity. Also, the number of anchor nodes in the network can be increased successively. Euclidean distance formula is used to measure the distance between general nodes and each fixed anchor nodes pair. This gives out the training dataset. The number of anchor nodes is kept fixed at 3 for simplicity and better result [8].

$$\log\text{sig}(m) = 1/\left(1 + \exp(-m)\right) \tag{2}$$

$$k = purelin(m) = m \tag{3}$$

For the hidden layer, log-sigmoid transfer function is used as shown in Eq. (2). For the output layer, linear transfer function is used as shown in Eq. (3).

V. Testing of the network

The trained network is further extended to predict the location of any number of sensor nodes on the same network grid. The network grid boundary can either be extended or reduced according to the expected area coverage. In that case the network needs to be retrained every time the change of area coverage is expected. Thus for the testing, the sensor nodes are deployed in a distributed manner throughout the grid and not compulsorily at the intersection points, only input values in the form of distance of those sensor nodes to the anchor nodes are fetched. Unlike in training, no target values are used here. The actual dataset is then compared with the output dataset to compute the localization error. Again, log-sigmoid transfer function is used for the hidden layer as referred in Eq. (2). For the output layer, linear transfer function is used as referred in Eq. (3).

Algorithm 1 Algorithm for Training

```
Input: Target data, distance between each of the sensor node
    with anchor nodes (3 anchor nodes are used here)
Output: Input to Hidden layer and Hidden to Output layer
    weight matrices
    Initialization: Initialize all the weights with the random
    values
 1: while the stopping condition not satisfies do
 2:     Calculate the net input using $I_m = \sum W_{km} * O_k + \theta_m$
 3:     Apply the activation function, $O_m = f(I_k)$
        Propagate the error in backward direction
 4:     Calculate the error for output layer using, $Err_m = O_m(1 - O_m)(T_m - O_m)$
 5:     Calculate the error for hidden layer using, $Err_m = O_m(1 - O_m)\sum Err_t * W_{mt}$
 6:     Update the weights and $\theta$(bias) values using,
        $\Delta W_{km} = l * Err_m * O_m$
        $W_{km} = W_{km} + \Delta W_{km}$
        $\Delta\theta_{km} = l * Err_m$
        $\theta_{km} = \theta_{km} + \Delta\theta_{km}$
        Check for the Stopping conditions
 7:     if ($\Delta W_{km} <$ threshold) OR (total number of iterations
        exceeds a certain predefined value) then
 8:         Stop
 9:     end if
10: end while
```

Algorithm 2 Algorithm for Testing

```
Input: Weights and bias produced during training phase,
    location of sensor nodes placed at random locations
Output: Target locations with minimum localization error
    Initialization: Initialize the required variables
 1: Calculate the distance between each of the sensor node
    with the anchor nodes
 2: for each hidden and output layer unit do
 3:     Calculate the net input $I_m = \sum W_{km} * O_k + \theta_m$
 4:     Apply the activation function, $O_m = f(I_k)$
 5: end for
 6: Calculate the Testing Error
```

VI. Experimental results

A. Simulation Settings

The following metrics are used for experimental setup:

- Number of anchor nodes $= 3$
- Number of generic sensor nodes $= 10$ (50 for graphical representation)
- Area of coverage $= 100 \times 100$
- Number of neurons in the hidden layer $= 10$
- Dimension of Input-Hidden matrix $= 3 \times 10$
- Dimension of Hidden-Output matrix $= 10 \times 2$

B. Input during Training

Table I represents the inputs provided to the algorithm. Target data represents the considered $2D$ coordinates of the nodes and the input data is the distance between sensor nodes and the anchor nodes.

TABLE I
Input Training Set

Target Data		Input data		
X_i	Y_i	$AN1$	$AN2$	$AN3$
		$(100, 0)$	$(100, 100)$	$(0, 100)$
0	0	100.00	141.42	100.00
10	0	90.00	134.54	100.50
20	0	80.00	128.06	101.98
30	0	70.00	122.07	104.40
40	0	60.00	116.62	107.70
...	...	...	...	...
90	100	100.50	10.00	90.00
100	100	100.00	0	100.00

C. Output after Training

Table II and Table III are formed after execution of the training algorithm. It consists of the weight values which will be used further for the generation of the estimated node locations.

D. Data collected after testing of one set of coordinate range

Table IV contains the actual location of the sensor nodes and estimated locations produced after execution of the testing algorithm. It also shows the error and average deviation generated from the difference of actual and estimated locations.

Here, $X_d = |X_A - X_e|$ and $Y_d = |Y_A - Y_e|$

$$Err = \sqrt{X_d^2 + Y_d^2}$$

$$Avg\ Deviation = \frac{X_d + Y_d}{2}$$

$$Avg\ Err = \frac{TotalError}{Number\ of\ Sensor\ nodes\ tested} \qquad (4)$$

From Table IV, we obtain average error (by Eq. 4) as follows:

$Total\ Err = (4.75 + 2.28 + 2.88 + 2.60 + 4.85 + 2.36 + 1.37 + 2.29 + 15.17 + 10.80) = 49.35$

$$Avg\ Err = \frac{49.35}{10} = 4.935$$

Omitting the last two extreme nodes that are $(0, 0)$ and $(3, 7)$, we get

$$Avg\ Err = 2.9225$$

Thus, accuracy is almost 98% to 97%. The nodes at $(0, 0)$ or $(3, 7)$ behave as outliers because they are more than 90% away from all of the anchor nodes individually. For instance, the node at $(0, 0)$, has the following distances (using Euclidean distance formula 1) from the anchor nodes:

Dist. from $AN_1 = 100$ units

Dist. from $AN_2 = 141.42$ units

TABLE II
WEIGHT MATRIX OF INPUT TO HIDDEN LAYER

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
1	-5.1180	-6.4990	-3.6620	-5.6500	-1.6370	3.7966	-14.5800	0.9715	-2.8650	-0.0190
2	-5.9560	-3.4700	0.1006	-9.2350	-1.9590	-8.3460	-28.0600	-1.7930	-3.4420	-14.2100
3	-0.0800	-15.9300	-2.0820	2.6343	-1.0240	-5.3510	0.6407	-2.3070	-6.1540	-2.0970

TABLE III
WEIGHT MATRIX OF HIDDEN TO OUTPUT LAYER

	1	2
W_1	0.9826	2.7836
W_2	-4.8062	0.7432
W_3	5.0985	-0.9141
W_4	6.6481	2.5808
W_5	4.7842	1.0690
W_6	4.0204	2.1161
W_7	-9.7163	0.7285
W_8	-1.7312	1.9019
W_9	0.2455	1.7532
W_{10}	-5.3959	-2.9658

TABLE IV
ERROR ESTIMATION TABLE

Actual Location		Estimated Location		Difference value		Err	Avg Deviation
X_A	Y_A	X_e	Y_e	X_d	Y_d		
78	99	81.2529	102.4664	3.2529	3.4664	4.75	3.36
66	96	67.3046	97.8720	1.3046	1.8720	2.28	1.58
68	104	70.8100	104.8196	2.8100	0.8196	2.88	1.71
70	95	71.2023	97.3001	1.2023	2.3001	2.60	1.75
85	94	87.5741	98.1068	2.5741	4.1068	4.85	3.34
77	88	74.9269	89.1192	2.0731	1.1192	2.36	2.52
66	92	65.2593	93.1466	0.7407	1.1466	1.37	1.20
68	107	69.9278	105.7550	1.9278	1.2450	2.29	1.59
0	0	9.0953	12.1464	9.0953	12.1464	15.17	10.62
3	7	10.7982	14.4666	7.7982	7.4666	10.80	7.63

Dist. from $AN_3 = 100$ units
$X_d = |X_A - X_e| = |0 - 9.0953| = 9.0953$
$Y_d = |Y_A - Y_e| = |0 - 12.1464| = 12.1464$
Hence,

$$\begin{aligned} Avg\ Deviation &= \frac{X_d + Y_d}{2} \\ &= \frac{9.0953 + 12.1464}{2} \\ &= 10.62 \end{aligned}$$

Thus, the results deviate by around 6% to 10% in such cases; however, with uniform distribution of anchor nodes over the grid area, the deviation converges in the range of 2% to 5%. The mean difference for both X and Y coordinates, denoted by E_X and E_Y, are calculated as shown below.

$$E_X = \frac{(\sum_{i=1}^{N_i} X_{d_i})}{N_i}$$

$$E_Y = \frac{(\sum_{i=1}^{N_i} Y_{d_i})}{N_i}$$

For the data sets obtained in Table IV, we get $E_X = 3.27367$ units and $E_Y = 3.56887$ units.

Applying the same technique on 50 random coordinate datasets, it was observed that throughout any location over a specific grid area (here 100 by 100) the accuracy rate is quite consistent(referring Fig. 1 and Fig. 2).

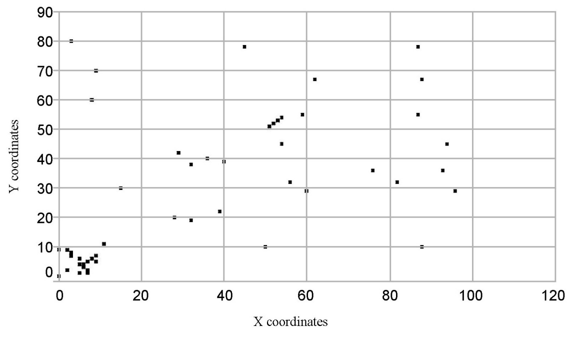

Fig. 1. Graph against the actual location of nodes.

E. Data collected after testing against multiple set of coordinate ranges

Fig.1 represents a simple, scattered point graph where each point on the $X - Y$ plane is the graphical location of sensor nodes. The dots represent the practically acquired coordinates on a particular floor-planning. For experimental purposes of our model, 50 such locations have been taken into consideration which is absolutely random or may be based on the sensing task accounted for. It is known that the grid size is 100 by 100 and the three anchor nodes are situated at the three extreme corners that are $(100, 100)$, $(0, 100)$ and $(100, 0)$. Thus the points close to these corners are also considered close to the relative anchor nodes. Moreover, the points near to the origin are also farthest from all anchor nodes. It is also clear from the figure that clustered position of sensor nodes are tested.

VII. CONCLUSION

*T*he experiment is based on practical observation and can be reconstructed on any platform. The authors have not used any already deployed tool. It is also observed that increasing the number of neurons in the hidden layer improves the performance of the network as well but to a certain extent.

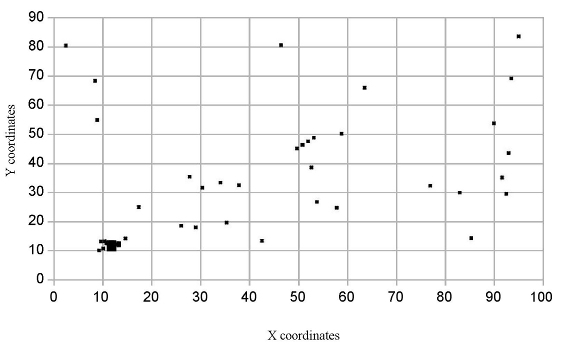

Fig. 2. Graph against the estimated location of nodes.

From Fig.1 and Fig.2 the output variations for X and Y coordinates can be easily visualized respectively. Thus, this makes it an efficient and effective localization algorithm in the application in a wireless sensor network. The main objective of developing this model is that it can be used for indoor tracking where GPS fails. The designed model can be effectively implemented with minimum cost and effort. Due to minimal hardware requirements, this model can be easily incorporated within bigger projects such as underwater acoustic system, underground seismic detectors, sensing based cyber systems, etc. In future, it can replace automated tracking devices requiring high profile image processing units, as this methodology does not require any image capturing device and only relies on sensing efficiency. Moreover, since location estimation is achieved with high accuracy, replacing GPS based systems for indoor tracking, floor planning, indoor spying, etc. is just a matter of time.

Acknowledgement

This work has been partially supported by the Project IN17MO07 “Formal Specification for Secured Software System”, under the Indo-Italian Executive Programme of Scientific and Technological Cooperation.

References

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

[2] J. Wang, R. K. Ghosh, and S. K. Das, “A survey on sensor localization,” *Journal of Control Theory and Applications*, vol. 8, no. 1, pp. 2–11, 2010.

[3] D. E. N. Bulusu, J. Heidemann, “Gps-less low cost outdoor localization for very small devices,” *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, 2000.

[4] S. Zhang, M. Er, B. Zhang, and Y. Naderahmadian, “A novel heuristic algorithm for node localization in anisotropic wireless sensor networks with holes,” *Signal Processing*, vol. 138, pp. 27–34, 2017.

[5] S. Zaidi, A. Assaf, S. Affes, and N. Kandil, “Accurate range-free localization in multi-hop wireless sensor networks,” *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3886–3900, 2016.

[6] P. Prasithsangaree, P. Krishnamurthy, and P. Chrysanthis, “On indoor position location with wireless lans,” in *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2002.

[7] X. Chen, L. Du, and Y. Fu, “Adaptive location algorithm based on bpla for wireless sensor network,” in *World Automation Congress (WAC)*. IEEE, 2012.

[8] S. Bhardwaj, “Ann for node localization in wireless sensor network,” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 2, no. 5, pp. 1724–1731, 2013.

[9] S. Souhila and M. Samira, “Positioning of node for static wsn,” in *International Conference on Advanced Wireless, Information and Communication Technologies (AWICT 2015)*. Elsevier B.V, 2015.

[10] E. Eugenio and A. Cortesi, “Wifi-related energy consumption analysis of mobile devices in a walkable area by abstract interpretation,” in *Distributed Computing and Internet Technology.ICDCIT 2017.Lecture Notes in Computer Science*, vol. 10109. Springer, Cham, 2017, pp. 27–39.

[11] S. Haykin, *Neural Networks and Learning Machines*. Prentice Hall, Cambridge, Massachusetts, USA, 2008.

[12] A. Chatterjee, “A fletcher-reeves conjugate gradient neural-network-based localization algorithm for wireless sensor networks,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 2, pp. 823–830, 2010.

[13] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Elsevier, Waltham, Massachu-setts, USA, 2011.