Zone-based Indoor Localization using Neural Networks: a View from a Real Testbed

Nafisa Anzum, Syeda Farzia Afroze, and Ashikur Rahman
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh
Email: {1005091.na, 0416052038.sfa, ashikur}@cse.buet.ac.bd

Abstract—Precise indoor localization is of great importance to automatically track people or objects indoors and plays a vital role in modern life. Despite a number of innovative research present in the literature indoor localization still remains an open problem. To trace the main reason we identify that in the present literature the tendency is to pinpoint the exact coordinates of a target device although most of the location based services (LBSs) do not require exact coordinates. To support LBS, one can simply divide the area of interest into several zones and perform “zone-fencing”, i.e., find under which zone the user is currently located at. In this paper, we propose a zone-based indoor localization scheme using neural networks. With the results from real world indoor settings, we show that a number of empty clusters is generated when the traditional counter propagation network (CPN) is applied as is. But a slight modification to the CPN reduces the number of empty clusters significantly and provides promising accuracy. The proposed scheme outperforms “k-Nearest Neighbor algorithm” (k-NN) and its promising accuracy makes it suitable for real-world deployment.

I. INTRODUCTION

With the ever-increasing popularity of smart-devices, people are becoming highly accustomed to Location Based Services (LBS) such as guiding someone into unknown places like shopping malls, hospitals, university campuses etc. Applications running over smart devices often need capability to track people or objects and monitor certain point-of-interests either in outdoor or indoor settings. Interestingly, an average person spends more time indoors than outdoors in a day. According to the Environmental Protection Agency (EPA) [1], the average American spends 93% of their life indoors. 87% of their life is truly indoors, then another 6% in automobiles. That boils down to only 7% of their entire life outdoors which is (surprisingly) only half of a day per week. Consequently localization is more active field of research in indoor settings rather than outdoor environment. Moreover, by virtue of Global Positioning System (GPS), going to unknown places, locating things anywhere in outdoors has become easier. However, the GPS is known to perform poorly in the indoor environment due to the signal attenuation caused by the construction materials. In addition, the interpretation of accuracy is not the same in outdoors and indoors. A five meters inaccuracy in outdoors may not affect that much because it still might indicate the same street or avenue but a five meters inaccuracy in indoors might indicate a different room or even a different building.
is easy to guide the user to the appropriate destination.

In this paper, we propose a zone-based solution to indoor localization to deal with the problem of uncertainty of exact location estimation. We assume that the area of interest is covered by multiple APs and is subdivided into several zones. While devising a solution at first we collect signal samples at a certain zone which are set of measured signal powers received from all reachable APs in different time. Then, we identify signal vectors which are the vectorial representation of the collected signal samples. Within a zone signal vectors taken at one location tend to be similar with signal vectors taken at nearby locations within the zone boundary, and differ in two locations located under two different zones. The zone of a new location is identified by pattern matching the signal vector at new location with the previously collected vectors.

The major contributions of the paper are summarized below:

- We propose a zone-based solution to indoor localization problem using Counter Propagation neural Network (CPN). The key challenge is that for noise and other reasons, the zones are not completely separable. Therefore, a straightforward application of CPN is not enough to predict a zone on the basis of sampled signal vectors. A simple modification to CPN solved the ambiguity among zones. The solution offers better time and space complexity and provides improved performance.
- We demonstrate that the basic CPN achieves 90% accuracy whereas the modified CPN provides 91% accuracy. The 8% misclassified data assigns the target device to adjacent zones and only 1% are totally misclassified.
- The proposed method outperforms the competitive k-Nearest Neighbor (k-NN) algorithm.

II. RELATED WORKS

A good number of research works on indoor localization has been carried out in the past decade. All techniques can be broadly classified into triangulation based [4], Received Signal Strength Indicator (RSSI) based [4, 5], Radio Frequency Identification (RFID) [6, 7] based, and neural network based approaches [2, 8–10].

The work presented in [6] develops LANDMARC which is a prototype that uses RFID technology for indoor location sensing. Such system strictly requires RFID tags which makes it unsuitable for many location based applications. Moreover, the proposed system incurs high latency. The system described in [7] also uses RFID technology. Additionally, it requires navigation devices and navigation servers which creates significant latency in determining indoor positions.

RSSI based techniques integrate fingerprinting method for indoor localization. Bolliger et al. [4] describes a fingerprint-based system which trains the system with the help of usage profiles that largely depends on user collaboration. Salazar et al. [5] presents indoor localization using the Wi-Fi RSSI fingerprinting based on Fuzzy Inference System estimator. Wi-Fi fingerprinting creates a radio map of a given area based on RSSI data from several APs and generates a set of RSSI data for a given zone location. However, this system requires an additional wearable wristband which incurs additional cost.


III. SYSTEM OVERVIEW AND PROBLEM FORMULATION

In this section we provide system architecture and formal description of the problem that we solve in this paper.

A. System overview

We use traditional fingerprinting approach to solve the zone-based indoor localization problem. Suppose the area of interest is a building. Each floor of the building can be viewed as a two-dimensional lattice which can further be divided into zones. Like other fingerprinting approach, the zone-based localization also consists of two phases: (i) an offline profiling phase, and (ii) an online localization phase.

Profiling Phase. The profiling phase creates a fingerprint database of signal vectors. A smart device (SD) is taken to every zone in the building and a certain number of signal samples are captured standing roughly around the centroid of a zone. The signal vectors representing the signal samples are recorded along with the zone number and sent to a central database. Note that, in a non-zone based approach the fingerprinting requires an exhaustive survey of all possible positions to build a radio map. This also requires a significant effort to measure and record the exact $x, y$ coordinates of the position on which the signal sample is collected. On the other hand, zone-based approach only needs to record the zone numbers instead of exact coordinates for a signal sample. Thus, a significant gain on processing cost, time and effort can be achieved using zone-based profiling.

Localization phase. Localization or positioning is basically location tracking that is performed in real time. In the localization phase, a smart device collects real-time measurements of RSS and compare it with the available fingerprint database using a search/pattern matching algorithm. The result of the algorithm shows the most probable zone of the smart device. Next subsection formally defines these two phases.

B. Problem formulation

Suppose we are given an area of interest $A$ that is divided into a set of $m$ zones $Z = \{z_1, z_2, \ldots, z_m\}$. The area $A$ is fully covered by a set of $n$ Wi-Fi stationary access points.
\begin{equation*}
AP = \{ ap_1, ap_2, \ldots, ap_n \}. \text{ Suppose, each zone } z_i \text{ is within the detectable range of at least three access points.}
\end{equation*}

**Fingerprinting:** Wi-Fi based fingerprinting is to collect and assign to each zone \( z_i \in Z \), a set of \( K \) Wi-Fi signal vectors at different time instances, \( V_{z_i} = \{ V_{z_i,t} | 1 \leq t \leq K \} \), and \( V_{z_i,t} = \{ ap_1 : rss_1, \ldots, ap_n : rss_n \} \) where \( rss_i \) is the received signal strength (RSS) value of \( ap_i \) observed by a smart device at zone \( z_i \). The number of access points sensed could vary at each scan, and from zone to zone. If an access point \( ap_i \) can not be sensed either in a scan or within a zone, signal strength for that \( ap_i \) is assigned to a minimum possible value \( rss_{min} \).

**Indoor localization:** Given a new Wi-Fi signal vector \( V_o = \{ ap_1 : rss_1, \ldots, ap_n : rss_n \} \) observed by a user’s smart device, indoor localization determines user’s probable zone by measuring the similarity between \( V_o \) and each of the Wi-Fi signal vector \( V_{z_i,t} \) at each zone \( z_i \)’s fingerprint data set \( V_{z_i} \).

\section*{IV. METHODOLOGY}

In this section we present two zone-based techniques.

\textbf{A. Distinct Access Point(s) Based Solution (DABS)}

While analyzing the sample data in our test bed, we observe that not all access points are active at all zones. Therefore, each zone can be uniquely identified by a set of access points only available at that zone. The number of access points to consider is an important issue. The simplest but less accurate approach is to look into the closest access point and use it to identify a particular zone. However, the coverage area of a single access point could be quite large and may include multiple zones. Thus, the accuracy is a major concern as an access point’s transmission range is about 150 feet while operating at 2.4 GHz and is about 50 feet while operating at 5 GHz band. With only two access points alias might be created. Consequently, we use three access points to uniquely identify a zone. These three access points are different for different zones.

While selecting three access points for a zone, it is intuitive to select those three that provide strongest signals compared to other APs in that zone. However, we took a different approach. At first we experimentally observed the fluctuations of signals coming from different access points on a zone. For example, Fig. 1 shows the RSSI values of different APs found within a certain time interval at the zone CSE-204. There were seven access points sensed in that zone. Notably, the stronger signals fluctuate more rapidly than weaker signals. While choosing three access points we select those three that provide non-overlapping signals or signals with minimal overlaps. The detail algorithm is given next.

At the profiling stage, the RSSI values of different access points (APs) found at each zone are stored. Recall that for each zone we need to collect \( K \) samples. The number of sample instances of each AP for this zone is also calculated. If this number is less than 50% of the expected instances \( K \), then it is marked as a weak AP for this zone and is eliminated from further consideration. For each of the valid APs sensed in a zone, the average and standard deviation of \( K \) samples are calculated. Let us denote these values at zone \( z_i \) as follows:

\begin{equation*}
AVG(z_i) : (ap_1 : rss_{avg_1}, ap_2 : rss_{avg_2}, \ldots, ap_k : rss_{avg_k})
\end{equation*}

\begin{equation*}
STD(z_i) : (ap_1 : rss_{std_1}, ap_2 : rss_{std_2}, \ldots, ap_k : rss_{std_k})
\end{equation*}

Next, all the valid APs are sorted in descending order of their RSSI level. After sorting, the first access point \( ap_f \) in the list is chosen for the zone as it provides the strongest signal, and its \textit{range} is calculated using the following equation:

\begin{equation*}
RANGE(z_i, ap_f) = rss_{avg_i} + rss_{std_i}
\end{equation*}

The rest of the APs from the list are checked one by one in order, and the first AP whose average signal strength does not fall within the range of the selected AP is chosen next. Similarly, the third AP selected is the one whose signal average does not fall within the ranges of first two already selected APs. The algorithm is summarized in Algorithm 1.

For localization, the algorithm needs the list of present APs and their RSSI values at the current location of the user. Then the set of distinct APs of each zone is examined one by one. If the three APs to identify a zone are all present at the current location, then a distance is obtained. The zone with the lowest vector distance is the detected zone for the current location of the user. Algorithm 2 describes the procedure.

The distinct AP based solution works well if all the access points show almost constant behavior all the time, that is, for a specific zone, the RSSI values of an access point are within certain range. Also for consecutive zones, three chosen APs could be all same. Moreover, the RSSI values vary rapidly in real environments making the solution unreliable.

\textbf{B. Counter Propagation Neural Network Based Solution}

In this section we present a neural network based solution using \textit{counter propagation network} (CPN). The CPN is very popular due to its simplicity in calculations and strong power of generalization. Some other approaches such as back-propagation networks might also be used, but it is computationally expensive. The CPN has the ability to recognize unseen patterns. In case of no perfect match, CPN provides closest match and for multiple closest match it provides composite output. The pattern matching is done in parallel against all previously learned vector patterns.
Algorithm 1 Identify three distinct APs for a zone

Require:
\( \text{zone} \): The zone for which we need to identify three APs

Data: RSSI values of APs found in that \( \text{zone} \)

Ensure:
\( \text{class} \): A list of APs that can be used to identify a \( \text{zone} \)

1: \( \text{APList} \leftarrow \emptyset \)
2: for all AP \( s \) found in \( \text{zone} \) do
3: \( \text{s.avg} \leftarrow \text{calculate average} \)
4: \( \text{s.std} \leftarrow \text{calculate standard deviation} \)
5: \( \text{s.count} \leftarrow \text{count number of valid RSSI values found} \)
6: \( \text{APList} \leftarrow \text{APList} \cup \{s\} \)
7: end for
8: \( \text{expvalue} \leftarrow \text{expected number of valid RSSI value} \)
9: for all AP \( s \in \text{APList} \) do
10: if \( \text{s.count} < 50\% \) of \( \text{expvalue} \) then
11: Mark \( s \) as weak AP
12: Delete \( s \) from \( \text{APList} \)
13: end if
14: end for
15: \( \text{queue} \leftarrow \text{sort APs in} \text{APList} \) in descending order of their RSSI average values
16: \( \text{upper_range} \leftarrow -\infty \)
17: \( \text{lower_range} \leftarrow \infty \)
18: \( \text{class} \leftarrow \emptyset \)
19: \( i \leftarrow 1 \)
20: while \( i \leq 3 \) do
21: \( \text{conflict} \leftarrow 0 \)
22: \( s \leftarrow \text{queue.pop}() \)
23: if \( s.\text{avg} \geq \text{lower_range} \) and \( s.\text{avg} \leq \text{upper_range} \) then
24: \( \text{conflict} \leftarrow 1 \)
25: end if
26: if \( \text{conflict} = 0 \) then
27: \( \text{class} = \text{class} \cup \{s\} \)
28: \( \text{upper_range} \leftarrow s.\text{avg} + s.\text{std} \)
29: \( \text{lower_range} \leftarrow s.\text{avg} - s.\text{std} \)
30: \( i \leftarrow i + 1 \)
31: end if
32: end while
33: return \( \text{class} \)

Algorithm 2 DABS Localization

Require:
\( \text{ClassList} \): an array of set of three APs that can be successfully used to identify a \( \text{zone} \)
\( \text{APList} \): List of sensed APs in current location with their respective RSSI average values

Ensure:
\( \text{zone} \): zone corresponding to the current location

1: \( \text{zone} \leftarrow \emptyset \)
2: \( \text{min_distance} \leftarrow \infty \)
3: for all zone \( z \) do
4: \( \text{class.ap} \leftarrow \text{ClassList}[z] \)
5: \( \text{distance} \leftarrow 0 \)
6: \( \text{found} \leftarrow \text{TRUE} \)
7: for all \( x \in \text{class.ap} \) do
8: for all \( y \in \text{APList} \) do
9: if \( y.\text{ssid} = x.\text{ssid} \) then
10: \( \text{distance} \leftarrow \text{distance} + (x.\text{avg} − y.\text{avg})^2 \)
11: else
12: \( \text{found} \leftarrow \text{FALSE} \)
13: end if
14: end for
15: end for
16: if \( \text{found} = \text{TRUE} \) then
17: \( \text{vector_distance} \leftarrow \sqrt{\text{distance}} \)
18: if \( \text{vector_distance} < \text{min_distance} \) then
19: \( \text{min_distance} \leftarrow \text{vector_distance} \)
20: \( \text{zone} \leftarrow z \)
21: end if
22: end if
23: end for
24: return \( \text{zone} \)

Before applying CPN, the entire testbed is divided into several zones and each zone is then assigned a unique gray code such that the code changes by one bit for each adjacent zones. For a zone, the adjacent zones are those that share at least one border with it, i.e., the zones located immediately above, below, left and right of it. The number of bits needed to represent a zone depends on the size of the testbed. If a testbed has \( n \) zones, the assigned gray codes will be of \( [\log_2 n] \) bits. These unique gray codes will be the output of the CPN.

Once the gray codes are assigned, we move on to the profiling stage. During profiling signal samples containing the signal strengths measured from each of the \( n \) access points are collected at each zone. Each signal sample is an \( n \)-dimensional vector collected at time \( t \) and is denoted by \( V_{zi,t} = (rss_1, rss_2, \ldots, rss_n) \). While collecting signal samples, the data collection device is put near the center of the zone and left out of the borders. The purpose of collecting samples in this way is to obtain training vectors representing a zone to be clearly separable from the training vectors collected (in the same way) in other zones.

A signal vector is augmented with the recorded zone numbers and constitute Signal Zone Vector (SZV):

\( SZV_i = \langle V_{zi,t}, z_i \rangle = (rss_1, rss_2, \ldots, rss_n, z_i) \)

From the characteristics of Wi-Fi RSSI values in ideal situation, we can deduce the following two observations:

(i) For a zone \( z_j \), \( SZV_j \) remains the same or varies within a fixed range.

(ii) For two zones \( z_i \) and \( z_j \), \( SZV_i \) and \( SZV_j \) are different.

From the above two observations, we deduce that despite some overlaps in WiFi signals, two zones can be easily distinguished if clusters can be formed corresponding to their vectors.

Design of the CPN. A feed-forward CPN generally consists of three layers as shown in Fig. 2. The first layer is the input layer. The total number of neurons in the input layer is equal to
Algorithm 3 Training Kohonen network

1: Initialize weight vectors of each output neuron with sample mean vectors for each location
2: repeat
3: for each vector $V_{z,i} = (rss_1, rss_2, ..., rss_n)$ do
4:   for $j = 1$ to $m$ do
5:     calculate $VAL_j = \sum_{i=1}^{n} rss_i \times w_{ij}$
6:   end for
7: end for
8: Assign the input vector to cluster $L_j$ for which the value of $VAL_j$ is maximum
9: for $i = 1$ to $n$ do
10:   $w_{ij}(new) = w_{ij}(old) + \beta (rss_j - w_{ij}(old))$
11: end for
12: until $N^{th}$ iteration

Training Kohonen Layer Weights. In the training phase we train the network so that the weight vectors are adjusted based on the training vectors and act as a centroid of each cluster. Unlike traditional CPN where the weight vectors are initialized randomly, we initialize weight vectors with a mean value of few samples taken from the training data set. For that purpose, in each zone $z_i$ at first we identify the associated neuron at the Kohonen layer and the weights connected with that neuron are assigned with the mean value of few samples collected at that zone. As learning progresses the weight vector ultimately gets aligned with the centroid of all samples collected at that zone. During training process the input vectors are applied one by one and for an input vector, $(rss_1, rss_2, ..., rss_n)$ the weight adjustment is made as follows. For each neuron $L_j$, $j = 1, 2, 3, ..., m$ of Kohonen layer, the output $VAL_j$ is calculated using the following equation:

$$VAL_j = \sum_{i=1}^{n} rss_i \times w_{ij}$$

(1)

After calculating all the values, the neuron having the maximum value is declared as the winning neuron for the current input vector. The weights connected with the winning neuron $L_j$ is updated using the following formula:

$$w_{ij}(new) = w_{ij}(old) + \beta (rss_j - w_{ij}(old))$$

where, $i = 1, 2, ..., n$. When a vector having the similar type of signal vector is applied to the input layer, the same neuron is going to fire and the same weights are updated. But for a vector of different zone, a different neuron gets fired and different weights get updated. Algorithm 3 shows the details.

Training Grossberg Layer Weights. When the input vector is applied to the input layer the corresponding zone number in gray code is also supplied. The weights at Grossberg Layer connected with the winning neuron $L_j$ are only updated. Suppose the function $Y_k(z_i)$ returns the $k$-th bit of the gray code assigned to zone $z_i$ for which the input vector was collected. The weights connected with neuron $L_j$ are trained using the following equation (for $k = 1, 2, ..., \log_2 m$):

$$g_{jk}(new) = g_{jk} + \beta (Y_k(z_i) - g_{jk})$$

Localization. Once the CPN is trained the localization becomes a simple mapping of the input vector to an output gray code. To detect a new location, the available APs and their corresponding RSSI values are recorded first. From the perceived signal strengths at the new location, an $n$-dimensional signal vector is created and applied to the input layer of the trained CPN. The output of each neuron $L_j$ at the Kohonen Layer is calculated using Equation 1. Then the firing neuron at Kohonen Layer is detected and the output of Grossberg Layer is produced. Thus the mapping from the input vector to the zone number of the new location is deduced.

Empty Cluster Problem. The Wi-Fi signals are extremely overlapping. Therefore, when a cluster initializes its centroid randomly, the algorithm cannot distinguish consecutive zones. As a result two or more consecutive zones are merged into one
leaving other centroids far away. These other centroids produce empty clusters. To deal with the empty cluster problem, we provide an initial hint of the position of different zones at the training phase rather than initializing the cluster centroids randomly. The initial centroids were calculated by taking the mean vectors of few sample data of each zone. Using this approach, the efficiency of the algorithm increased and empty clusters dropped down to a very few.

V. EXPERIMENTAL RESULTS

In this section we present a detail experimental result obtained from a real testbed.

A. Experimental setup and technical details

As a testbed we use the WLAN infrastructure of New Academic Building, Bangladesh University of Engineering and Technology (BUET). This is a 12-storied building. The busiest floors are from first to fourth floor because most of the classrooms and laboratories are located there. Four floors we found 24 access points, more or less evenly dispersed. The floors of the building are of similar structures. We (virtually) divide each floor into 8 zones giving us a total of $4 \times 8 = 32$ zones. While creating zones we only consider the open public spaces of the building such as the corridors and lobby and omit the office rooms and class rooms because those were private inaccessible zones. The floor map of the first floor and the division of zones are shown in Fig. 3. Zones are shown in yellow colors. Also each zone is assigned a gray code. As we have 32 zones, a 5 bit gray code was needed.

Data has been collected in two phases—training and testing. In training phase 400 data samples were collected at each zone. For testing purpose, another 100 data samples per zone were collected. Thus, we had a total of $400 \times 32 = 12,800$ vectors for training and $100 \times 32 = 3,200$ vectors for testing purposes.

The training data was used to train the weight vectors of the CPN. Since the testbed consists of 24 APs, each of the 12,800 training vectors had 24 components. The sensed RSSI values range over $-40$dBm (strongest) to $-100$dBm (weakest). While scanning at a specific zone, it may happen that some APs are not sensed in the scan due to AP’s limited transmission range. It was observed that on an average $6 \sim 8$ different AP’s signal were sensed at each scan from each zone. The vector components for non-reachable APs were set to a very small RSSI value of $-130$dBm.

B. Performance metrics

We consider the followings metrics for comparison:

- **Correctly Classified**: the percentage of test data that has been classified to their correct zones.
- **Adjacent Classified**: the percentage of test data that has been classified to a zone adjacent to the actual zone.
- **Misclassified**: the percentage of test data that has been classified to a zone which is neither its correct zone nor adjacent to the actual zone.

C. Results

Once the CPN is trained, we apply test samples to see the accuracy of the model. When a test sample arrives, at first CPN determines the firing neuron and then it gets the gray code of the corresponding zone as output from the final layer. The CPN performs with great accuracy and generates a single firing neuron at Kohonen layer for each test data. Table I shows the normalized confusion matrix for all zones. Except for a few specific zones, all zones have nearly 100% correct classification accuracy. To be exact, 22 zones among 32 have 100% correct classification rate. Four zones have 96−99% and one zone has 90% correct classification rate. For four zones CSE107, CSE115, CSE209, and CSE213, though the correct classification rate is 83%, 29%, 31%, and 6% respectively, the incorrectly classified test data of those zones were classified to adjacent zones. The highest misclassification rate of 24% is found at zone CSE111. The model falsely predicted CSE119 as their zone which is not adjacent to CSE111. However, CSE119 is adjacent to CSE115 which in turn is adjacent to CSE111 (see Fig. 3). Thus, even falsely predicted or misclassified data also picks a nearby zone.

D. Comparison with k-NN algorithm

k-Nearest Neighbor (k-NN) [12] is a widely known algorithm used for indoor localization problem [3, 13]. We implement the algorithm and run with $k = 1, 3$ and 5. Testing of DABS, CPN and k-NN method is done on the same test data set. The results on the test data is shown in Fig. 4. For k-NN with $k = 1$, the percentage of correctly classified data is 83%, adjacent classified is 11% and misclassified is 6%. For
k-NN with $k = 3$, correctly classified, adjacent classified, and misclassified percentages are 86%, 9%, and 5% respectively. For k-NN with $k = 5$, those rates are 88%, 8%, and 4% respectively. Thus, it is easy to see that when the value of $k$ is increased correct classification accuracy increases, adjacent classification and mis-classification decrease. However, the accuracy is lower compared to CPN and DABS. For DABS correct classification rate is 91%, adjacent classification rate is 4%, and misclassification rate is 5% whereas for CPN correct classification rate remains the same but misclassification rate drops down to only 1% with slight increase in adjacent classification rate to 8%.

VI. CONCLUSION AND FUTURE WORK

We present two zone-based indoor localization techniques namely DABS and modified CPN. Both techniques outperform traditional k-NN algorithm. k-NN is not a pragmatic solution as its performance largely depends on volume of the sample data and consumes a large memory. Also k-NN is a lazy learner as it has a high time complexity but the proposed algorithm performs quickly by checking test vector against learned vectors in parallel. Although the correct classification rate is same for DABS and CPN, CPN outperforms DABS in misclassification rate. In future, we plan to address multiple access point failures. Also finding optimal zone size remains an open problem.

ACKNOWLEDGMENT

This research work was conducted at the department of Computer Science & Engineering, Bangladesh University of Engineering & Technology (BUET). We acknowledge BUET for its generous support to make this work publishable.

REFERENCES


