Grid-Based Spatial Index Method for Location-Based Nearest Neighbour Search

Aung Zaw Myint and Khin Mo Mo Tun

***Abstract*—Geo-spatial data becomes more and more large amount of data on the Web. On the other hand, managing massive spatial data is one of challenges for supporting spatial queries and high performance computational is also needed to support spatial queries. Thus there is needed to solve this criteria is to create a better spatial indexing method. The proposed method is to create Grid-based R-tree index structure for k nearest neighbour query and range query. R-tree is constructed with Minimum Bounding Rectangle (MBR) that contains a group of objects. The proposed system is combined R-tree with grid index that is reduced overlapping and covering area. The proposed system is to support spatial queries efficiently and also supports speed up computational performance.**

***Index Terms*—K nearest neighbor search, R-tree, grid-index, LBS.**

I. Introduction

The increasing of inter-network technologies and Internet of Things (IoTs) devices has become commonly useful for consumers on their mobile smart aware devices. Thus the quality of spatial data is needed to support on their real-time location via location-based services (LBS). Google Maps Services and Social networking services (SNS), such as Twitter and Facebook are provided online users to reveal their location during emergencies, retrieve information about nearby restaurants and hotel, and acquire local traffic. The current growth of LBS services, companies is boosting their efforts to automatically locate consumers for advertising and marketing purposes. LBS software services are also considered for mobile applications to represent a new business sector and E-commerce. Meanwhile, retrieval of k-Nearest Neighbour (k-NN) efficiently and quickly requires an informative and effective index structure that can reduce the search space. Spatial index is used by spatial databases to optimize spatial queries.

Spatial databases are fully-fledged databases that, in addition, enable the storage, retrieval, manipulation, querying, and analysis of geometries like points, lines, and retrieval, manipulation, querying, and analysis of geometries like points, lines, and regions representing, for example, the geometries of cities, roads, and states respectively.

In order to handle spatial data efficiently, database management system needs an index mechanism that will output the data quickly according to their current location. There are many index structures such as K-D-B tree works only in points data, B- tree, R-tree, Hilbert-Curve tree, BSP-tree and Quard-tree. Most spatial database application uses R-tree indexing method because it is the most widely accessed method [1].

This study proposes a new algorithm to speed up the performance of k-Nearest Neighbour (*k*-NN) retrieval on spatial database. The proposed method named as R-tree based grid index is a hybrid index structure of R-tree and grid indexing technique [2].

Grid index is used for locating objects. As grid index is easy in implementations such as updating and creating index, it is simple and efficient way of spatial indexing. The grid index extracts only locations from nearest indices and sends theses indices to R-tree. R-tree is used to retrieve nearest objects [3].

Additionally, the remainder of this paper is organized as follows. In Section II, we describe related works of this paper. In Section III, we describe background theory. In Section IV, we discuss the overview of proposed system. In Section V, we explain computing grid index. In Section VI, we describe construction of R-tree. In Section VII and VIII, we discuss our expected experimental and then conclude the paper.

II. Related Works

Processing k-Nearest Neighbour (KNN) query based on location has been well studied in spatial database. R-tree method was proposed by Guttman in 1984 in order to handle spatial data efficiently, as required in computer aided design and geo-data applications and to extract the objects by current location [4], [5]. Zhang *et al*. proposed a grid cell based continuous k-NN query processing method (CkNN) [6].Inverted file-*R** tree and *R** tree – inverted file are geo-textual indices that loosely combine the R* tree and inverted file [7]. Hariharan, B. Hore, C. Li,S. Mehorotra [8] proposed the *KR**-tree structure that captures the join distribution of keywords in space and significantly improves performance. The query performance of grid index is robust while updating positions of the objects [9]. The proposed index used Distributed grid index from server transmission and clients examine the received index with the unique ID number to each grid-cell [10]. Z. Li,K.C. K. Lee, B. Zheng,W.C. Lee,D.LLee,X [2] proposed IR-tree. This paper proposed to support efficient geographic document and IR-tree enables the pruning of textually and spatially irrelevant subsets.

As to k-NN query algorithm [3], its system has studied well in traditional databases. The idea of this system is to establish a static R-tree-like structure which is developed from Rtree. It cannot handle continuous queries and update

Manuscript received June 25, 2019; revised October 12, 2019.

Aung Zaw Myint is with Faculty of Computing, University of Computer Studies, Yangon (UCSY), Myanmar (e-mail: aungzawmyint@ucsy.edu.mm).

Khin Mo Mo Tun is with the Faculty of Computing, University of Information Technology, Myanmar (e-mail: khinmomotun@uit.edu.mm).

queries. DJ Oneil *et al*. X. Cao, G. Cong, Christian S. Jesen,Jun J. Ng,Beng C. Ooi, N.T.Phan,D. [7] proposed Spatial Web Object Retrieval System(SWORS) that is capable of efficiently retrieving spatial web objects that satisfy spatial keyword queries. This system use IR tree and inverted file for index. SWORS supports location-aware top-k text retrieval (LkT) query and the spatial keyword group (SKG) query that retrieves a group of objects that cover the query keywords.

III. Background Theory

The rapidly expanding technology of mobile communication will give mobile users capability of accessing information from anywhere and anytime. It is also a distributed system with a network to communicate between different machines. Wireless communication is needed to enable mobility of communicating devices. In recent years, Location-Based Service (LBS) is growing rapidly among mobile users. As the mobility is the most distinguishing feature of the mobile computing environment, location becomes an important piece of information for LBS. Therefore, spatial database for spatial locations has become an important area of people’s interest and research. A fundamental issue in this area is how to store and operate spatial data efficiently. Quickly executing k-Nearest-Neighbour (k-NN) and range queries in spatial database applications requires an informative and efficient index structure that can effectively reduce the search space.

A. R-tree Index Structure

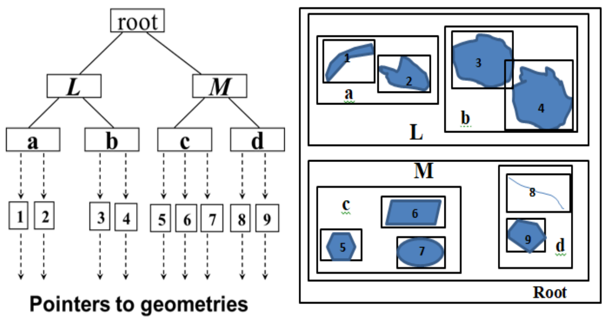
R-trees index method was presented by Guttman [1]. R-tree is an index tree-structure that uses multi-dimensional indexes. R-tree derived from B-tree. An R-tree is a depth-balanced tree in which each node corresponds to a disk page (i.e., the number of entries in each node is limited). R-trees are based on Bounding Box. Objects are entirely contained inside the bounding box. The actual objects are recorded in the leaves point that are enclosed minimum bounding rectangles (MBRs). This is also the reason for the benefits R-trees have in the matter of dynamic indexing. But the bounding boxes used in R-tree nodes can overlap.

The following is the description of R-Tree:

Let *M* be the maximum number of entries that will fit in one node and let *m*<=*M*/2 be a parameter specifying the minimum number of entries in a node. An R-tree satisfies the follow properties.

1) Every leaf node contains between m and M index records unless it is the root.
2) For each index record (I, tuple-identifier) in a leaf node, I is the smallest rectangle that spatially contains the n-dimensional data object represented by the indicated tuple.
3) Every non-leaf node has between m and M children, unless it is the root.
4) For each entry (I, child-pointer) in a non-leaf node, I is the smallest rectangle that spatially contains the rectangles in the child node.
5) The root nodes have at least two children unless it is a leaf.
6) All leaves appear on the same level.

The following figure is to illustrate this situation for a simple two-dimensional structure. Fig. 1 shows R-tree structure.



Fig, 1. R-tree structure.

B. Insert Algorithm in R-tree

Insert a new index entry *E* into R-Tree *T*.

```
Insert (E, t)
1. L ← ChooseLeaf (E, t)   > select a leaf node L where
   to place E
2.      If L need not split
3.    then install E
4.    else SplitNode(L)
5.  AdjustTree(L)


Choose Leaf(E, t)
1.  N←t
2.  while N is not a leaf
3.    do choose the entry F in N whose rectangle FI
          needs least enlargement to include EI
4.        N←FP
5.  return N

AdjustTree(L)
1.  N←L
2.  if L was split previously
3.     NN←resulting second node
4.  While N is not the root
5.     do P←parent(N)
6.          En←entry in P which points to N
7.          Adjust EnI so that it tightly encloses all entry
            rectangles in N
8.          if N has a partner NN which results from an
            earlier split.
9.              then create Enn so that EnnP point s to NN
            and EnnI enclose all rectangles in NN.
10.                 if there is room in P
11.                    then add Enn to P
12.                   else splitNode(P) to produce P and
            PP
13.         N←P
14.         if there exists PP
15.         then  NN← PP
16. Return
```

C. Search Algorithm in R-tree

Search algorithm accomplishes the following task, given an R-Tree whose root node is T, find all index records whose rectangles overlap a search rectangle S. An entry in a node as E(EI, EP), where EI represents the smallest rectangle bounding the sub-tree or the spatial object, EP is the pointer to the sub-tree or the spatial object.

```
SearchSubTree(t, s)
1. If t is not a leaf
2.   then for each entry E in t do
3.        if EI overlaps S
4.        then SearchSubTree(EP, s)
5.   else SearchLeaf(T, s)

SearchLeaf(t, s)
1. for each entry E in t
2.      do if EI overlaps s
3.           then output E
```

We can apply the searching of an R-tree to find objects that overlap a search object, say o, by the following steps.

```
SearchObj(t, o)
1.        s←bounding box of the search object o
2.   SearchSubTree(t, s)

and revise the above SearchLeaf(t, s) as follows
SearchLeaf(t,s)
1.   for each entry E in t
2.        do if EI = s
3.             then if EP = o
4.                  then output E
```

D. Delete Algorithm in R-tree

Remove an index record E from an R-Tree.

```
Delete(E, t)
1.        L ← FindLeaf(E, t)
2.   If  L is null
3.      Then return
4.   Remove E from L
5.   CondenseTree(L)
6.   If the root node has only one child.
7.      then make the child the new root.

FindLeaf(E, t)
1.   if t is not a leaf
2.      then for each entry F in t
3.              do if FI overlaps EI
4.                   then FindLeaf(E, FP)
5.      else  for each entry F in T
6.              do  if FI = EI & FP=EP
7.                   then return T

CondenseTree(L)
1.   N ← L
2.   Q ← empty
3.   while N is not the root
4.      do P ← parent(N)
5.         En ← the entry of N is P
6.         if N had fewer than m entries
7.              then delete En from P
8.              add N to set Q
9.         else  adjust EnI to tightly contain all entries in
           N
10.      N←P
11. Reinsert all entries of nodes in set Q according to
           their level.
```

E. Grid Index Structure

Grid is a pre-defined partitioning index in which the region is partitioned into rectangular cells because it indicates the pre-defined spatial region. Grid covers a part of the space that object position is inside the boundaries of a grid cell and this object belongs to this cell. Šidlauskaset al [9] proposed the grid index which is one of the principal framework index structures for moving objects database. The performance of grid index commonly uses in multi-dimensional queries. The grid index file can be stored entirely in memory. The storage structure of a grid index file stores the size parameters *m* and *n* that is block pointer of the grid. Then index files stores the buckets of the grid. Grid is a 2-d array that every cell in the array matches to a square cell with a length of grid cell size. Every grid cell in the array has a link to a list of buckets which contains the data object [9]. Each bucket has bucket size bs and metadata fields where metadata contains the next bucket, the number of entries and pointer to the next bucket in Fig. 2.

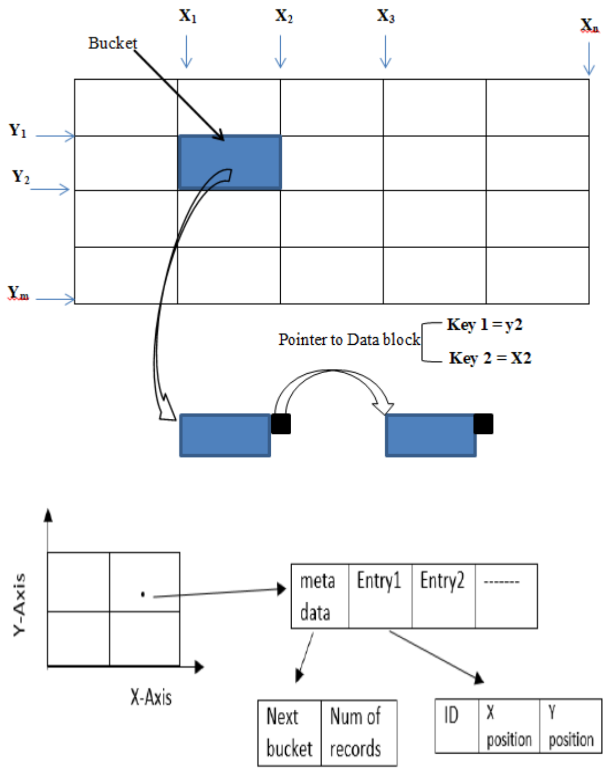


Fig. 2. Grid index structure.

IV. Proposed System

The proposed system is explained in detail. This system offers k nearest neighbour results to the user based on their current location quickly by using R-tree based grid indexing technique.

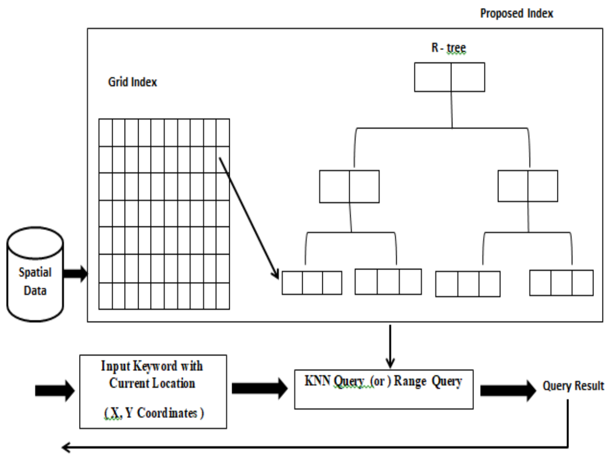

Proposed Index
R - tree
Grid Index
Spatial Data
Input Keyword with Current Location (X, Y Coordinates)
KNN Query (or) Range Query
Query Result

Fig. 3. System overview.

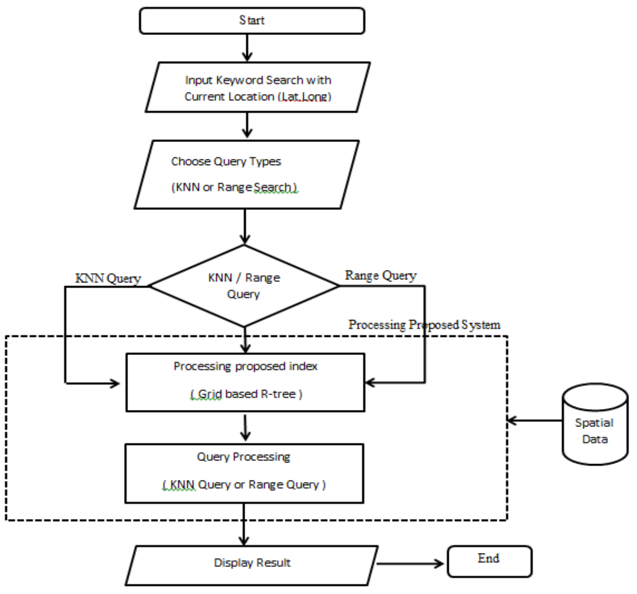

Start
Input Keyword Search with Current Location (Lat,Long)
Choose Query Types (KNN or Range Search)
KNN / Range Query
KNN Query
Range Query
Processing Proposed System
Processing proposed index (Grid based R-tree)
Query Processing (KNN Query or Range Query)
Spatial Data
Display Result
End

Fig. 4. Flow chart of proposed system.

First of all, locations of objects (latitude and longitude values) can be acquired via GPS in user device. The objects are located in two dimensional grid cells in Figure 3. Then, it identifies coordinates of the top left corner and right bottom corner within specified area. We collect clinics, mini-marts, restaurants, shops, and stores within above coordinates. Then index for each collected location and current location are computed. Thus nearest indices around index of current location can be acquired. The grid index extracts only locations from nearest indices and sends these indices to R-tree in Fig. 4. Therefore, the usage of large memory space in R-tree can be reduced. In R-tree, nearest indices that are sent by grid index are constructed as a tree. R-tree is composed of root node, intermediate node and leaf node. R-tree sort active branch list (ABL) by ordering MINDIST. ABL is a list that calculates the distance of objects.

V. Computing Grid Index

The grid index is composed by grid cells. Each cell represents a region of space generated by partitioning the domain using a uniform grid, which can then be assigned unique identifiers and used for spatial indexing purposes. It uses coordinates of objects and sorts them into grids, where grids have their own identifier index for faster querying. It is simple and efficient way of spatial indexing. A grid index file is organized into a two dimensional structure that is shown in Fig. 5. The geographically related data (i.e.: points that are close to each other) are stored in the same data block. The grid index cannot represent objects, it can only present points. In fact, the only kind of index that can handle where am I queries: is given by a location (i.e., coordinate), and then find the objects that contains the location. Therefore, performance of grid index for range queries is to find objects that are located within a certain range and such performance of grid index for nearest neighbor queries is to find the nearest neighbor of a data point.

```
Input latitude, longitude
Output grididx
Begin
LatIng ygntl = (17.136160,96.009705);
LatLng ygnbr = (16.7851054,96.1791551000);
latdiff = (ygntl.latitude- latitude);
londiff = (ytntl.longitude – longitude);
w = (latdiff / factor);
h = (londiff / factor);
grididx = (h * maxwidth ) + w;
End
```

Fig. 5. Grid index algorithm.

VI. Construction R-tree

R-tree can be used (nearest neighbour) search for some places. In R-tree, nearest indices that are sent by grid index are constructed as a tree. R-tree is composed of root node, intermediate node and leaf node. The processing step of the R-tree construction for nearest neighbour search is shown in Fig. 6.

```
begin
    if node is a leaf node
        Compute distance from p to all points contained in current node.

        Update the current best NNs and Current best distance if the distances
        between p and all points are less than the maximum distance of nearest
        NNs list.
    else
        Sort the MBRs contained in current node using MINDIST in ascending
        order.
        This sorted list is denoted the Active Branch List (ABL).
        Compute the minimum of the MINMAXDIST of all MBRs.
        Apply pruning strategy to the ABL to remove unnecessary branches.

        Iterate on this ABL until it is empty. For each iteration step, apply pruning
        strategy recursively to its children node.
        When the ABL is empty,
        Return nearest neighbor.
```

Fig. 6. Nearest neighbor search on R-tree.

In this algorithm, p is query point and points in a node are objects. MBR means Minimum Bounding Rectangle of each leaf node. The two ordering metrics in R-tree are MinDist and MinDist. MinDist is the distance of object O from query point

P. MinMaxDist is the minimum of the maximum possible distances to a face or vertex of the MBR containing O. MinDist and MinMaxDist offer a lower and an upper bound on the actual distance of object O from the query point P respectively.

There are two pruning strategies. They are downward pruning and upward pruning. In downward pruning, it allows pruning an entry from Active Branch List (ABL). If MINDIST of MBR [i] (length of ABL) is greater than MINMAXDIST, discard MBR[i] and all other nodes with greater MINDIST from ABL. In upward pruning, discard MBR[i] and all other nodes with greater MINDIST from ABL if MINDIST of MBR[i] is greater than current best distance.

The distance is calculated by using Haversine formula. This formula is used as follows.

Let lon1, lat1 be longitude and latitude of current location and lon2, lat2 be longitude and latitude of next location respectively.

dlon = lon2-lon1

dlat = lat2-lat1

a = (sin (dlat/2))^2 + cos(lat1)× cos(lat2)×(sin (dlon/2)) ^2

c = 2× atan2 (sqrt (a), sqrt(1-a))

$d = R \times c$

where, a=the square of half of the straight-line distance between the two points

c=the great circle distance in radians

d=the distance between the two points

R=radius of the earth (R=6371.01km)

Although Euclidean distance is suitable for most spatial datasets, computing the distance between two points on earth based on longitude and latitude is actually not very precise. Thus, Haversine formula is suitable in calculating distance for spatial object. After calculating distance, R-tree keeps the sorted buffer of at most nearest neighbour.

VII. Experimental Result

The performance evaluations of the proposed system on processing time, response time are covered. Moreover, the comparisons on evaluation results for the proposed indexing scheme, no indexing scheme, and traditional R-tree indexing scheme. In order to implement an efficient k-NN search application, need a structured framework. First of all, locations of objects (latitude and longitude values) can be acquired via GPS in user device. The objects are located in two dimensional grid cells. Then, it identifies coordinates of the top left corner and right bottom corner within specified area. We collect restaurants, ATM machines, hotels, hospitals, gift shops, cinemas within above coordinates. Then index for each collected location and current location are computed. Thus nearest indices around index of current location can be acquired. The grid index extracts only locations from nearest indices and sends these indices to R-tree. Therefore, the usage of large memory space in R-tree can be reduced.

In R-tree, nearest indices that are sent by grid index are constructed as a tree. R-tree is composed of root node, intermediate node and leaf node. R-tree sort active branch list (ABL) by ordering MINDIST. ABL is a list that calculates the distance of objects.

The system evaluated on client-server model. In client side, it is user level side for mobile devices while evaluating the system. It is necessary to create API key for Google map. Google map does not work in mobile client without creating API key.

Therefore, the user has to register in the Google APIs Console that user wants to use Google Maps for Android. After creating API key, Google maps API key is added inside the application element. When the required permission access, the user can see map in mobile client. If the user wants to see related information of nearest objects, API key of Google places must be created. Google Places API is one among the many APIs provided by Google and this is to get geographic information about places. By using Global Positioning System, this component may be accessed via GPS receiver to get the current location of the Smartphone devices. Android provides access to the above components to facilitate the implementation of LBS services

In server side, locations of objects are divided into dimensional grid cell. After computing index of each grid cell, send only indices nearest to the current location to R-tree. These indices are constructed into tree structure. After performing the processing steps in R-tree, the nearest neighbour will be retrieved. The server sends these nearest results to the client.

In the evaluation, the proposed indexing scheme can reduce the processing time rather than traditional R-tree method that is shown in Table I.

Moreover, k-nearest neighbour results can be given back quickly to mobile user by using the proposed indexing scheme in Fig. 7.

TABLE I: Excepted Results for Proposed System

Service Types	R-tree Index	Proposed Index
Restaurants	40	15
ATM Machines	30	12
Hospitals	55	20
Hotels	70	40
Cinemas	68	43
Gift Shops	77	34

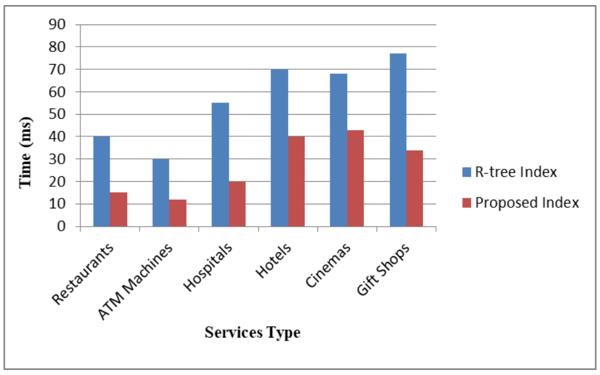


Fig. 7. Processing time of different indexing schemes.

VIII. Conclusion

Nearest neighbor search such as k-nearest neighbor query and range query based on user's current location is very important to get useful geo-information efficiently. A k-nearest neighbour query retrieves k objects in a given objects set which are closet to the query point q. Processing k-nearest neighbour query efficiently requires spatial indexing methods. In this paper, we proposed grid based R-tree indexing method system. The system can be used effectively the retrieval of user relevant geo-spatial information. The proposed system can reduce searching time and reduce unnecessary node visiting cost. The memory space can be reduced by using that proposed indexing techniques. The speed performance of this system outperforms the traditional R-tree system. As further work, we will consider on moving objects retrieval using the proposed index structure.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Khin Mo Mo Tun guided Aung Zaw Myint to do this research using data mining method. Aung Zaw Myint implemented the research by using hybrid method which combined grid data structure and R-tree index structure. Khin Mo Mo Tun approved system design and flow chart of this system. Aung Zaw Myint is writing this research paper and is testing the outcome results.

References

[1] A. Guttman, "R-Tree: A dynamic index structure for spatial searching," *ACM SIGMOD*, pp. 47-57, 1984.

[2] Z. Li, K. C. K. Lee. And Z. W. C. Lee "Ir-tree: An efficient index for geographic document search," *IEE TKDE*, vol. 3, no. 4, pp. 585-599, 2011.

[3] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of th top-k most relevant spatial web objects," *PVLDB*, vol. 2, no. 1, pp. 337-248, 2009.

[4] Y. Zhang, "The improvement and implementation of k nearest neighbor," *Computer Development and Application*, vol. 21, no. 2, pp. 18-21, 2008.

[5] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," pp. 656-665, 2008.

[6] N. Vincent, "Nearest neighbor queries," *ACM SIGMOD*, 1995.
W. Zhang, J. Z. Li, and H. W. Pan, "Processing continuous k- nearest neighbor queries in location dependent application," *International Journal of Computer Science and Network Security*, vol. 6, no. 3, March 2006.

[7] X. Cao, G. Cong, C. S. Jesen, J. Jun, N. T. Phan, and D. Wu, "SWORS: A system for the efficient retrieval of relevant spatial web objects," 2018.

[8] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (sk) queries in geographic information retrival(gir) systems," pp. 655-665, 2008.

[9] D. Šidlauskas, S. Šaltenis, C. W. Christiansen, J. M. Johansen, and D. Šaulys,"Trees or grids?: Indexing moving objects in main memory," in *Proc. thnternational Conference on Advances in Geographic Information Systems*, 2009, pp. 236-245.

[10] K. J. Park, "Location-based grid index for spatial query processing," *Expert Systems with Applications*, vol. 4, pp. 1294–1300, 2014.

Khin Mo Mo Tun was born in Yangon, Yangon Region, Myanmar in 1970. She received her master degree in information science in 1997 and the Ph.D (IT) degree in 2004 from University of Computer Studies, Y angon, Myanmar. She received her Ph.D degree in performance evaluation in high performace computing. She worked as an associate professor and the head in the Faculty of Computing in 2015 to 2017 at University of Computer Studies, Yangon. She is currently the head of the Faculty of Computing at University of Information Technology, Yangon. Her areas of research interests are digital image processing, data mining ,modeling and network security.

Aung Zaw Myint was born in Pakokku, Magway Region, Myanmar in 1982. He received his master in computer science degree from Moscow Institute of Electrical Engineering MIET), Russia in 2007. He is currently studying and doing research for doctor of philosophy (IT) in the Faculty of Computing, University of Computer Studies, Yangon. He is working for Ministry of Defence in IT section.