2014 28th International Conference on Advanced Information Networking and Applications Workshops

A First Look at Caching during TIP-tree Navigation for Location Based Services

Wendy Osborn
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
Email: wendy.osborn@uleth.ca

***Abstract*—This paper explores the issue of caching during TIP-tree navigation for location-based services. The TIP-tree is a recently proposed spatial access method for supporting continuous point queries in a tourist information system. In addition, its navigation strategy avoids repeating searching, thus providing efficient processing of user trajectories that are not known in advance. This paper proposes a caching strategy that further improves upon the efficiency of the TIP-tree navigation algorithm. A performance evaluation shows that significant improvement is possible when navigation and caching are applied together.**

***Keywords*—location-based services;spatial access method; continuous query processing; navigation**

I. Introduction

A location-based service [15] provides information to the user of a mobile device, such as an tablet or smartphone, based on their current location. For example, a user can issue a query to a server to find the location of the nearest few restaurants based on their current location. As the user moves around, however, the nearest few restaurants will change, and therefore be updated, on their device. Location-based services can serve many applications, such as tourist information services [13], [8], [4], [3], [11].

Location-based services require efficient continuous spatial query processing strategies [6]. A continuous spatial query involves the processing of a moving query, such as a moving point, nearest neighbours to a moving point, or a moving region query. One approach to processing a continuous spatial query is to perform a new query for every update on the user's trajectory. However, this results in high network and unnecessary processing costs [16]. So many approaches attempt to reduce the number of queries that need to be executed.

Several have been proposed that utilized spatial access methods [1], including [16], [17], [5], [7], [9], [12]. Limitations of these approaches include the need to repeated search the spatial index, caching a significant amount of data, or maintaining a sparse index. A recently proposed spatial access method, the TIP-tree [11], organizes point data and related items of interest, in a way that supports navigation, given a user trajectory, within the structure itself, and therefore repeated searching from the root is not required. In addition, the trajectory is not required to be known in advance.

The TIP-tree was shown to outperform existing spatial access methods with respect to disk accesses. However, one limitation of the TIP-tree is the repeated retrieval of the same node in sequence. If the user trajectory has many co-ordinates that are close together, then many nodes only need to be fetched from the server once, and that portion of the trajectory can be processed on the mobile device. Combined with the in-structure navigation, this leads to a significant reduction in the number of node accesses required from the server.

Therefore, this paper proposes an update to the TIP-tree navigation strategy that takes node caching on the mobile device into consideration. First, the data that needs to be cached is identified. Then, we identify the situations where the current cached data is still valid, and therefore a call to the server for updated data is not required.

The navigation strategy that utilizes caching is compared to the original (i.e not-caching) navigation strategy with respect to the number of disk accesses required to locate information for the user. It is discovered that when caching is applied, there is a significant reduction in the number of disk accesses required for retrieving information for the user.

This paper proceeds as follows. Section II presents some related work in the area of continuous query processing using spatial indexing. Section III presents a summary of the TIP-tree and its navigation algorithm. Section IV presents the motivation and approach to continuous point query processing in the presence of caching. Section V presents the evaluation of the proposed caching strategy. Finally, Section VI concludes the paper and provides some directions of future work.

II. Related Work

Some strategies exist for processing continuous spatial queries that utilize a spatial index such as an R-tree [2] or a modified grid file [10].

Song and Roussopoulos [16] propose an approach to continuous nearest neighbour query processing that utilizes existing stationary nearest neighbour approaches (e.g. [14]) to obtain a superset of nearest neighbour points, so that the result stays current while the query point moves around. An issue with this approach is in choosing an appropriate value of m, so that fewer query calls are made but not at the expense of significantly increased storage at the client.

Tao et al. [17] propose a strategy that utilizes an R-tree for speeding up searches. One limitation of this approach is that "vertical" searching for new co-ordinates must be performed repeatedly. An improvement to this approach was proposed by Park et al. [12], which proposes a "horizontal" search for

978-1-4799-2652-7/14 $31.00 © 2014 IEEE
DOI 10.1109/WAINA.2014.37

nearest neighbour objects along a trajectory. Although their strategy is shown to be efficient when spatial indexes are used, the entire trajectory must be known in advance.

A caching strategy is proposed by Hu et al. [5] to support multiple spatial query types, such as nearest neighbour and region. Their strategy caches on the mobile device results from previous queries and the R-tree nodes that lead to them. The cached partial R-tree is always searched first, with required additional objects that are not available in the cached R-tree fetched afterwards from the server. One issue is that the caching overhead and local processing may be costly.

Jung et al. [7] propose a continuous nearest neighbour approach using a grid index. Every cell in the grid contains a minimum bounding rectangle (MBR), which defines the bounds that contain the points in the cell. If a query does not overlap the MBR, none of its points will either. One limitation is that many portions of the index may be sparse, and thus space is wasted.

Lee et al. [9] propose a strategy that attempts to reduce the number of required queries by fetching both required and additional complementary objects, using an R-tree. One issue is that repeated searching is still required to obtain enough complementary objects.

III. Background

In this section, we present some background on both the TIP-tree and its navigation strategy.

The TIP-tree [11] is a spatial index that provides a user with access to items of interest, which are at a location that is close to the user, based on their current location. It has two key features:

- The TIP-tree provides location-based access to items that are stored within TIP itself, and stored externally to TIP in an information repository such as a digital library collection.
- The TIP-tree can be navigated from within the structure itself to continually present information to the user as the move around.

Figure 2 depicts an example TIP-tree structure and a navigation example (from [11]). Some of the TIP-tree structure will be summarized below, followed by the navigation example. More details on TIP-tree construction can be found in [11].

The root node is indicated with bold lines. The child node that is accessed from the NE location of the node represents a region of space that covers the North Island of NZ. From this node, the child node that is accessed from the NW location represents a region of space that contains Auckland and Hamilton.

The trajectory consists of locations in Auckland (point 1), Hamilton (point 2), somewhere in central North Island (point 3), and Wellington (point 4). An initial query is performed to locate the node containing points of interest (POIs) in Auckland, and send them to the user's mobile device. The user then proceeds to Hamilton. On arrival in Hamilton, the POIs for Hamilton are fetched and sent to the user's device. As the user proceeds south, the navigation strategy proceeds up one level of the tree to the parent of the node containing Auckland and Hamilton.

Figure 1 presents the navigation algorithm from [11]. In the algorithm, (x, y) is the set of co-ordinates that represent user query and X is the node whose region contains the query point. An initial search is done to locate a stating node, given the query point. From there, the navigation proceeds as follows. First, a new set of user query co-ordinates are received from the mobile device, and the current node X is fetched. Then, the one of the following situations occurs:

- The user query is in the region of space represented by X, and is in a region that contains a list node of items. In this case, the information is retrieved and sent to the user.
- The user query is in the region of space represented by X, and is in a region that contains a link to a child node. The query point is transferred to the child node of X.
- The user query is no longer in the region of space represented by X. The user query is sent to the parent node of X. If the existing node happens to be the root, then the search terminates.
- Otherwise, there is no further information to send to the user.

The navigation repeats through the above situations until it terminates.

IV. Caching for TIP-tree Navigation

In this section, the motivation for this work is presented, followed by the proposed approach to caching and navigation given the existing of caching.

Given the example summarized in Section III, and the experiments that were performed in [11], we observed the following: many points along the user trajectory may reside in the same region of space that is represented by a node. This is resulting in the same node being retrieved many times, when it is not necessary. This node (and the information on all items that are reference from it) can be cached on the mobile device, which will stop the occurrence of multiple retrievals for a sequence of trajectory points.

Referring back to Figure 2, we see that Auckland and Hamilton reside in the region of space that is reference by the same node. Therefore, the additional node retrieval to retrieve Hamilton could have been avoided. Similarly, one less node retrieval could have occurred for the last two points on the trajectory as the user made their way to Wellington.

Given a node (and information on all items that are referenced from it), we can cache this data on the mobile device and use it while it is still considered valid. Referring back to Figure 2, we see that the following information is referenced from a node:

- One or more list nodes that contain items of interest, and,
- One or more regions of space corresponding to a child node.

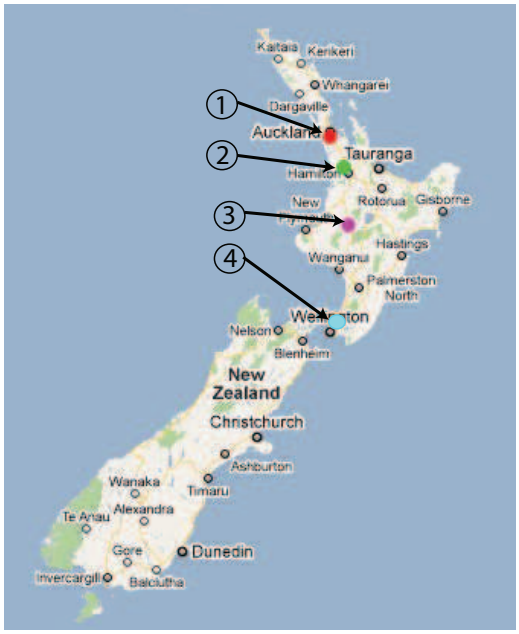
```
traversing = true;
search_point (x,y)
        = obtain_start_co-ordinates(x,y);
node X = search(x,y);

while(traversing) {

   (x,y) = new_co-ordinates(x,y);
   fetch_node(X);

   /*is search within the bounded region
     covered by current node?*/
   if( (x,y) within node_MBR(X) ) {
      Loc = determineLoc(X);

     /*does location contain coordinate?*/
     if(contains_co-ordinate(Loc)) {
         Sights = access_all_sites(Loc);
         display_all_sights(Sights);
     }

     /*traverse subtree?*/
     else if(with_node_MBR_of_child(Loc)){
         X = child_node(X);
     }
     /*otherwise, there is no information
        to display, so nothing is done
        until co-ordinates are updated.*/
  }

  /*if search outside current region,
  move to parent region if exists.*/
  else if( has_parent_node(X) ) {
     X = parent_node(X);
  }
  /*otherwise, outside of space covered by
    index, so terminate search*/
  else {
         traversing = false;
  }
}
```

Fig. 1. TIP-tree Navigation (from [11])

When this information is cached, it is considered valid under the following circumstances, and a query does not need to be performed on the server:

- The user query point does not move outside of the region of space that corresponds to the current node, and
- The user query does not move inside of the region of space corresponding to a child node that is referenced from the current node.

Otherwise, a query call needs to be made from the mobile device to the server. For the first circumstance above, at least the parent node will need to be retrieved, and possibly along with others in order to locate the node whose region contains the current query point. For the second circumstance, the child node will need to be retrieved.



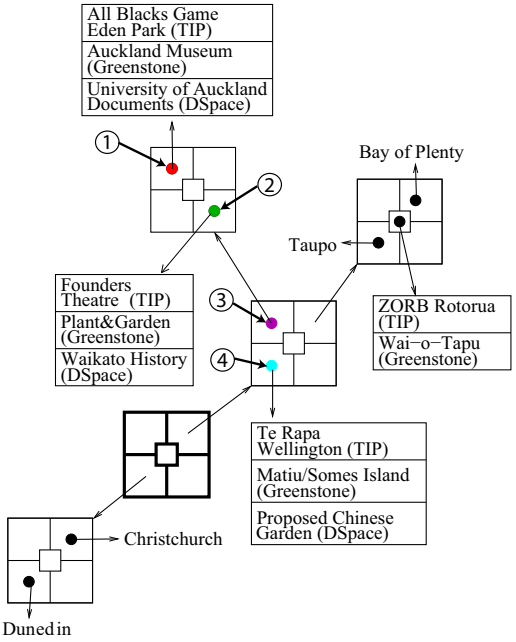


Fig. 2. TIP-tree with navigation example (from [11])

However, if a region of space is significantly large, and the user trajectory consists of points that are very close together, this will result in cached information remaining valid for many query points on the trajectory. This, in turn, will result in significantly fewer calls to the server for updated information.

If we refer back to Figure 2, we observe that only two node accesses are required now, instead of the four accesses that were required initially. The second node access only takes place when the user query moves south of Hamilton and ultimately moves outside of the region of space that is reference by the first node. The navigation strategy on the

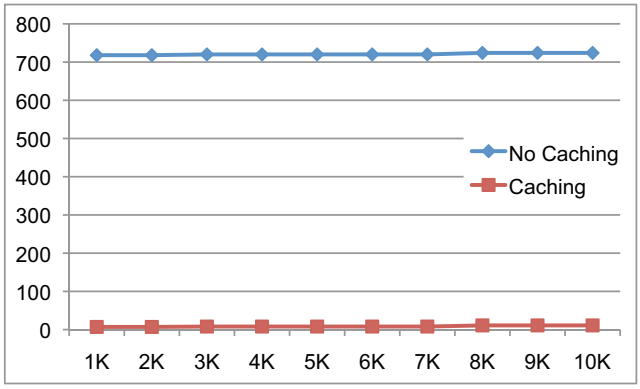


Fig. 3. Data Set Size vs. Disk Accesses

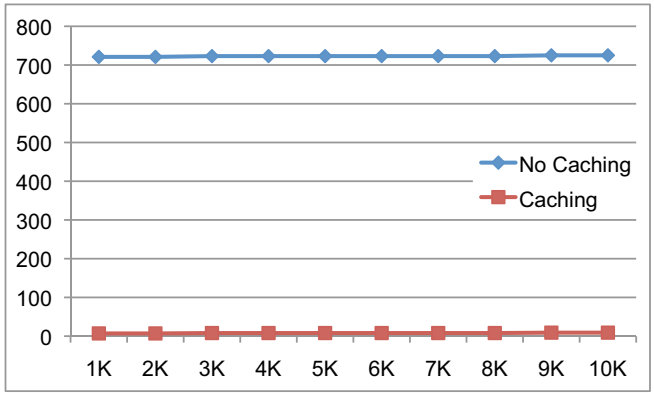


Fig. 4. Data Set Size vs. Disk Accesses (High Density)

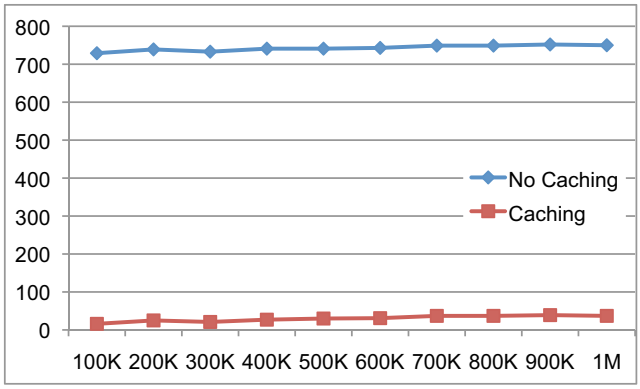


Fig. 5. Data Set Size vs. Disk Accesses (High Point Count)

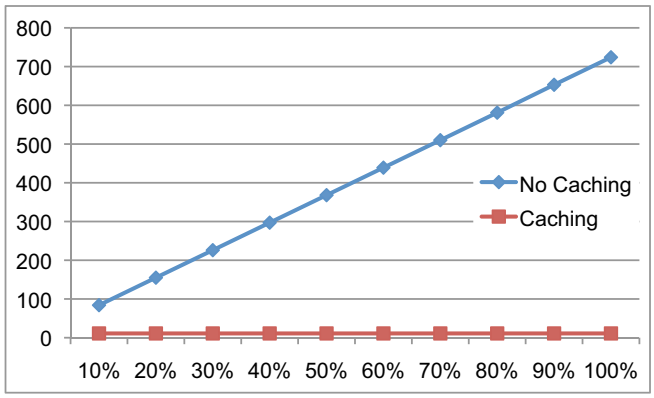


Fig. 6. Path Points vs. Disk Accesses

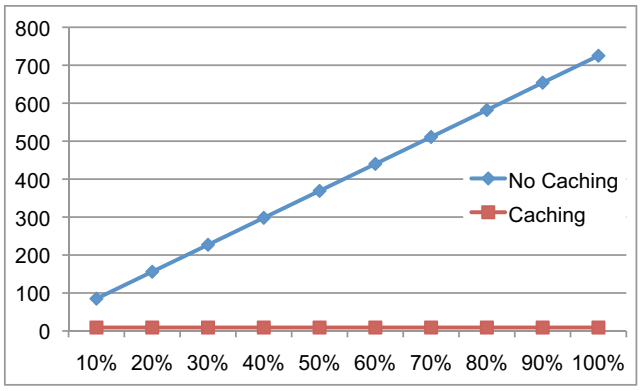


Fig. 7. Data Set Size vs. Disk Accesses (High Density)

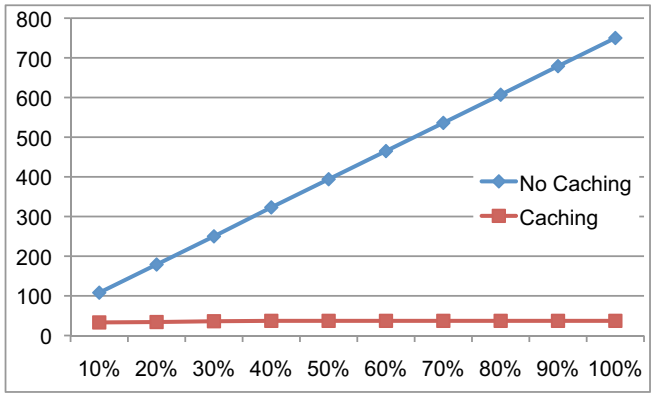


Fig. 8. Path Points vs. Disk Accesses (High Point Count)

server then proceeds to the parent node, and send that node (and all related information) to the mobile device.

V. Evaluation

This section presents the methodology and results of the performance evaluation. Here, we compare the performance of TIP-tree navigation when caching is utilized, versus TIP-tree navigation when caching is not used.

The evaluation uses four data sets - one user trajectory (T) and three sets of co-ordinates (P1, P2 and P3). The set T contains 712 co-ordinates that represent a user's trajectory between two cities on the North Island that are 16 kilometres apart. The trajectory was recorded using a mobile device. The set P1 consists of 10,000 co-ordinates, which represent various locations around the North Island of New Zealand. P2 consists of 10,000 co-ordinates, which represents locations from around the Waikato Region of New Zealand. Both P1 and P2 consist of co-ordinates that represent the location of actual municipalities in New Zealand, as well as randomly generated co-ordinates that represent other points of interest. The purpose of set P2 is to to be able to evaluate the strategies for a very dense point set. P3 consists of 1,000,000 randomly generated co-ordinates from around the North Island of New Zealand. The purpose of set P3 is to be able to evaluate the strategies for a very large point set.

The performance criteria for all tests is the number of disk accesses required in the TIP-tree for locating items of interest that are near the user query. For the navigation algorithm that does not utilize caching, we assume the worst-case scenario - a disk access is required every time a node is touched. When caching is used, we still record a disk access for every node touch. However, this only occurs when the cached node is no longer valid and the mobile device sends co-ordinates to the server to fetch the next node.

Our evaluation includes six tests. The first three tests examines the effect of the number of co-ordinates in the TIP-tree on the number of disk accesses required to locate items of interest for the user. The remaining three tests examines the effect of the number of co-ordinates on the user trajectory on the number of disk accesses needed for locating information.

A. Data Set Size

For the first test, we set P1 and trajectory T. For each of the 10 test runs performed, a TIP-tree was created using a subset of the points in P1, beginning with 10% of the co-ordinates, 20% of the co-ordinates, up to the entire set. Then, both the caching and non-caching versions of the navigation algorithm are executed using the entire trajectory T, and for both the number of disk accesses are recorded.

The second and third tests are similar to the first test. The only difference is that set P2 is used for the second test, while set P3 was used for the third test.

Figures 3, 4 and 5 depicts the results of the first, second and third test respectively. Over all tests, we see a common trend of a significantly lower number of disk accesses when caching is used over when caching is not used. In addition, we see for all tests that the number of co-ordinates in the index does not result in a significant change in the number of disk accesses required to process T. For test one, the number of disk accesses required for processing T when caching is used is between 7 and 11, while the number required for the non-caching scenario is between 718 and 724. For test two, between 7 and 9 disk accesses were required in when caching was used, versus between 721 to 725 when caching was not used. For test 3, between 16 and 39 disk accesses were required when caching was used, versus between 729 and 750 when caching was not used. Another observation at this point is that the number of disk accesses when caching was used increases when the number of co-ordinates in the index increases - as noticed in test three where the number of co-ordinates in the index is into the hundreds of thousands. However, this increase is very minimal when compared to the index size.

B. User Trajectory Size

For tests four, five and six, we vary the number of points on the user trajectory T, while keeping the number of points in the index constant. For the fourth test, we create a TIP-tree with using all co-ordinates in set P1. For each of the 10 tests runs, we use 10% of the co-ordinates on T, followed by 20% of the co-ordinates, up to all of the co-ordinates on T. For all tests runs, the overall distance of 16 kms is maintained.

Again, the fifth and sixth tests are similar to the fourth test, with the only difference being that set P2 (i.e a dense 10,000 co-ordinate TIP-tree) is used for the fifth test and P3 (i.e. a 1,000,000 co-ordinate TIP-tree) is used for the sixth test.

Figures 6, 7 and 8 depicts the results of the fourth, fifth and sixth tests respectively. Again, we see some common trends among all of the tests. First, the number of disk accesses required when caching is used is still significantly lower than the number of disk access when caching is not used. Second, when caching is not used, the number of disk accesses increases linearly as the number of points on the trajectory increases. However, the surprising result is that the number of disk accesses appears to remain constant when caching is used and does not appear to be affected by the number of points on the user trajectory!

For the fourth test, the number of disk accesses required for processing T when caching is used is constant at 11, where that number varies between 84 and 724 when no caching is used. For the fifth test, the number of disk accesses required when caching is used is constant at 9, while the number varies from 85 to 725 when no caching is used. Finally for the sixth test, the number of disk access when caching is used varies from 33 to 37, and from 108 to 750 when no caching is used. Again, we do see that for a larger number of co-ordinates in the index, the number of disk accesses required when no caching is used is higher than when when fewer points exist in the index. However, this number is still constant and significantly lower than when caching is used.

VI. Conclusion

In this paper, we propose an update to the TIP-tree navigation strategy that takes node caching on the mobile device into consideration. The data that needs to be cached is identified first. Then, we identify the situations where the current cached data is still valid, and therefore a call to the server for updated data is not required.

The navigation strategy that utilizes caching is compared to the original (i.e not-caching) navigation strategy with respect to the number of disk accesses required to locate information for the user. It is discovered that when caching is applied, there is a significant reduction in the number of disk accesses required for retrieving information for the user. Furthermore, we find that when caching is considered, a constant number of disk accesses is achieved, regardless of the number of points on the same-length user trajectory.

Some directions of future research include the following. The first is to extend the navigation strategy to work for continuous k-nearest neighbour and region queries. Another is to investigate support for moving co-ordinate data, as currently this work assumed stationary co-ordinates. In addition, not all items of interest are point data, and therefore investigating the inclusion of items of non-zero size is important. Finally, further evaluation of the navigation with caching strategy with longer trajectories is required.

REFERENCES

[1] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30:170–231, 1998.

[2] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pages 47–57, 1984.

[3] A. Hinze and Q. Quan. Trust- and location-based recommendations for tourism. In *Cooperating Information Sytsems (Coopis)*, pages 414–422, 2009.

[4] A. Hinze, A. Voisard, and G. Buchanan. TIP: Personalizing information delivery in a tourist information system. *Journal of IT & Tourism*, 11(3):247–264, 2009.

[5] H. Hu, J. Xu, W. Wong, B. Zheng, D. Lee, and W.-C. Lee. Proactive caching for spatial queries in mobile environments. In *Proc. 21st Int't Conf. on Data Engineering*, 2005.

[6] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Comput. Surv.*, 42(3):12:1–12:73, 2010.

[7] H. Jung, S.-W. Kang, M. Song, S. Im, J. Kim, and C.-S. Hwang. Towards real-time processing of monitoring continuous k-nearest neighbour queries. In *Proc. 2006 Int'l Conf. Frontiers of High Performance Computing and Networking*, 2006.

[8] P. Klante, J. Krshe, and S. Boll. Accessignts - a multimodal location-aware mobile information system. In *Proc. 9th Int'l Conf. on Computers Helping People with Special Needs*, Paris, France, July 2004.

[9] K. Lee, W.-C. Lee, H. Leong, B. Unger, and B. Zhang. Efficient valid scope computation for location-dependent spatial queries in mobile and wireless environments. In *Proc. 3rd Int'l. Conf. Ubiquitous Information Management and Communication*, 2009.

[10] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Trans. Database Syst.*, 9(1):38–71, Mar. 1984.

[11] W. Osborn and A. Hinze. Tip-tree: A spatial index for traversing location in context-aware mobile systems to digital libraries. *Pervasive and Mobile Computing*, 2013. in press.

[12] Y. Park, K. Bok, and J. Yoo. An efficient path nearest neighbour query processing scheme for location-based services. In *Proc. 17th Int'l Conf. Database Systems for Advanced Applications*, 2012.

[13] S. Poslad, H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf. Crumpet: Creation of user-friendly mobile services personalized for tourism. In *Proc. IEE 3G2001 Mobile Communication Technologies Conf.*, London, UK, 2001.

[14] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. *SIGMOD Rec.*, 24(2):71–79, May 1995.

[15] J. H. Schiller and A. Voisard, editors. *Location-Based Services*. Morgan Kaufmann, 2004.

[16] Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *Proc. 7th Int'l Symp. on Advances in Spatial and Temporal Databases*, pages 79–96, 2001.

[17] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *Proc. 28th Int'l Conf. Very Large Data Bases*, pages 287–298, 2002.