2014 IEEE 15th International Conference on Mobile Data Management

RoadEye: A System for Personalized Retrieval of Dynamic Road Conditions

Anirban Mondal, Avinash Sharma, Kuldeep Yadav, Abhishek Tripathi, Atul Singh, and Nischal Piratla
{anirban.mondal, avinash.sharma, kuldeep.r, abhishek.tripathi3, atul.singh, nischal.piratla}@xerox.com
Xerox Research Centre India
Prestige Technology Park II
Marthahalli, Bangalore - 560103, INDIA

***Abstract*—Awareness of dynamically changing road conditions is crucial for a safe and quality driving experience, as well as, in augmenting trip planning. This work addresses the problem of keeping users informed in a *timely* and *personalized* manner about road conditions arising from both scheduled and ad hoc events. We propose RoadEye, a system for *personalized* retrieval of dynamic road conditions. The key contribution of RoadEye is the ψR-tree, which is a novel R-tree-based index augmented with linked lists for facilitating quick and personalized retrieval of user-queried road conditions. Our performance study indicates that the ψR-tree is indeed effective in retrieving dynamic road conditions with reduced query response times and disk I/Os.**

I. Introduction

Awareness of dynamically changing road conditions is crucial for a safe and quality driving experience, as well as, in augmenting trip planning. For example, when a driver suddenly encounters a deadly pothole or a fallen rock, it may lead to vehicle damage or even a fatal accident. The seriousness of this issue is further exacerbated in case of poor lighting conditions (at night, during fogs, street lights that are dysfunctional) and difficult weather conditions (tropical rains, snowfall or storms).

Incidentally, considerable changes in road conditions can occur due to scheduled works (e.g., road maintenance work, speed bumps at strategic locations, public processions etc.) or as a consequence of *ad hoc* events (e.g., accidents, potholes, water-logging on roads due to sudden rainfall or broken water pipes, obstructions due to snowfall etc.). In the former case, it may be easier to keep commuters informed about the locations and the nature of obstructions (e.g., the height of a speed bump). However, in the latter case, the ad hoc nature of the events makes it extremely challenging to keep commuters informed in a timely manner. Thus, an end-to-end system for keeping commuters informed about ad hoc road conditions in near real-time settings becomes imperative.

Consider a user Alice, who wishes to visit *Museum of Fine Arts, Boston*. As depicted in Figure 1, the current location of Alice is tagged as "My Location" and her destination is tagged as DOI ("Destination of Interest"). We shall henceforth use the term 'DOI' to refer to a spatial region of interest pertaining to any given user. In this example, we can intuitively understand that Alice needs to be able to visualize various planned road works as well as ad hoc events around her DOI, as depicted by the red color ellipse in Figure 1. Such *context-aware visualization* would facilitate her in effectively planning the trip with adequate safety considerations. It would also help in dynamic decision-making by Alice e.g., in case she has to deviate from a planned route due to road congestion caused by an accident. In addition to visualization of road conditions, the dashboard interface shown in Figure 1 would also enable Alice to receive *personalized alerts* as well as to pose *specific questions* about her DOI (e.g., about the availability of public parking slots) which can be answered by other users.

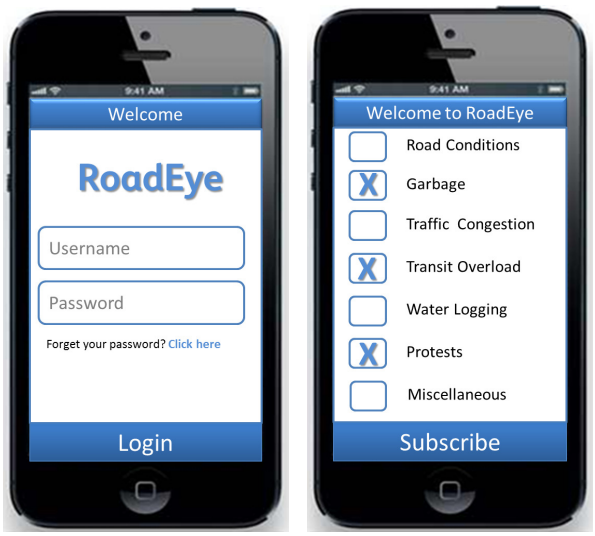
Intuitively, the notion of road conditions is inherently subjective i.e., the perception of the severity of a given road condition may vary considerably across users depending upon their preferences. For example, a user Alice may prefer a well-lit road albeit with frequently-spaced speed bumps and some relatively small potholes. In contrast, another user Jack may have low preference for the same road because he generally dislikes speed bumps and even small-sized potholes. Moreover, it is possible for the same user to have varied perceptions about the condition of the same road based on temporal considerations. For example, Alice may have low preference for an otherwise high-quality road during peak hours of traffic such as office hours. Thus, context-aware visualization of road conditions also needs to be *personalized* depending upon the preferences of each user.

Figure 2 presents two screenshots of the RoadEye mobile application. RoadEye provides a unique login credential to each of its users (see Figure 2a) so that they can access personalized information about road conditions from their mobile devices. Each user can subsequently set her event preference configuration, as depicted in Figure 2b. This enables a given user to visualize road conditions in a personalized and context-aware manner. Furthermore, each user may have personal preferences about the event updates that she would like to receive. These preferences keep changing based on her context (i.e. travel plan, weekday, etc). For instance, Alice may be interested in traffic congestion information. On the other hand, Bob could be interested in dynamic events such as road accidents, water-logging and so on. Moreover, for a given weekday, Alice may be interested in receiving event updates related to transit overload. RoadEye mobile application provides an option to subscribe to specific event types as shown in Figure 2b. If a user chooses to receive a

978-1-4799-5705-7/14 $31.00 © 2014 IEEE
DOI 10.1109/MDM.2014.42

Fig. 1: RoadEye dashboard for context-aware & personalized visualization of road conditions (Map courtesy: Microsoft Bing)

specific event type update, a push notification is sent to her mobile device whenever a new event of that type is registered on the RoadEye system.



(a) (b)

Fig. 2: RoadEye Mobile Application: Screenshot of login screen and event preference configuration

An efficient spatial indexing mechanism becomes a necessity to enable such personalized visualization of dynamic road conditions. While spatial indexing has motivated several research efforts [4], [5], [7], [8], [10]–[13], [15], [16], none of these approaches facilitate in effectively retrieving information about road conditions for a given spatial region in a *personalized* manner. In a similar vein, existing systems, such as Nericell [9], FixMyStreet [1] and Ushahidi [2], also lack personalization.

This work addresses the problem of keeping users informed in a *timely* and *personalized* manner about road conditions arising from both planned as well as ad hoc events. The main contributions of this work are three-fold:

1) We propose RoadEye, a system for *personalized* retrieval of dynamic road conditions.
2) We propose the ψR-tree, a novel R-tree-based index augmented with linked lists for facilitating quick and personalized retrieval of user-queried road conditions in RoadEye.
3) Our performance study indicates that the ψR-tree is indeed effective in retrieving dynamic road conditions with reduced query response times and disk I/Os.

The remainder of this paper is organized as follows. Section II provides an overview of existing works. Section III describes the system architecture of RoadEye. Section IV presents our proposed ψR-tree index. Section V reports the performance evaluation. Finally, Section VI concludes the paper with the directions for future work.

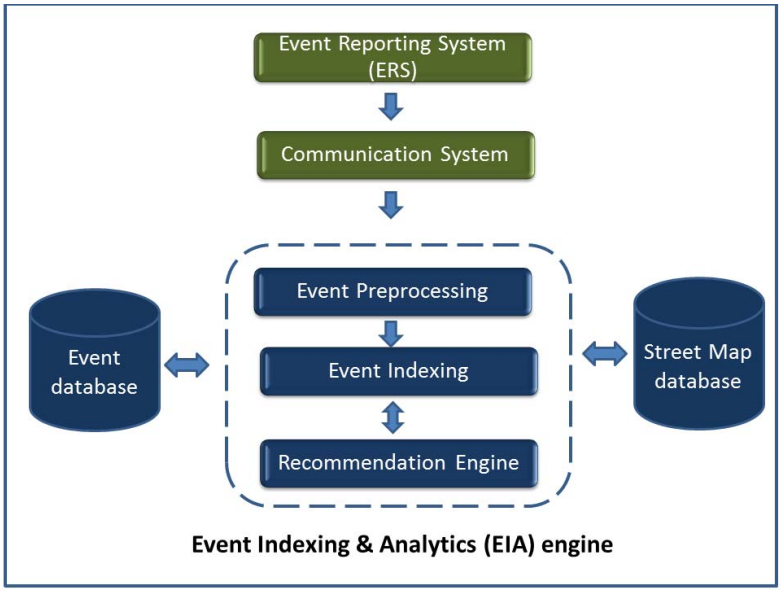


Fig. 3: Architecture of the RoadEye system

II. Related Work

The problem of spatial indexing has motivated several research efforts. In this regard, the R-tree [5] is one of the most popular spatial index structures. Given its popularity, several variants of the R-tree have been proposed over the past three decades, the most notable ones being the R^+-tree [13] and the R*-tree [4]. R-tree-based indexes have also been proposed for traditional distributed domains such as clusters e.g., the dR-tree [10], the M-Rtree [7] and the MC-Rtree [12]. Furthermore, R-tree-based indexes, which are specifically targeted towards the spatio-temporal domain, include the Spatio-Temporal R-tree (STR-tree) and Trajectory-Bundle tree (TB-tree) [11], the time-parameterized R-tree (TPRtree) [16], Lazy Update R-tree (LUR-tree) [8], and the Multiversion 3D R-tree(MV3R-tree) [15]. A good survey on spatial (and spatio-temporal) indexing can be found in [3].

Notably, despite the significant amount of research on R-tree-based indexes, the ψR-tree differs significantly from the existing approaches. First, unlike other R-tree-based indexes, it augments the nodes of the R-tree with linked lists for associating various types of road conditions with different regions in space. Second, it keeps the linked list at each node of the R-tree sorted in descending order of importance to enable quick retrieval of the types of road conditions that are more prevalent in a given region. Third, it facilitates personalized retrieval of data concerning road conditions based on user preferences. Thus, to the best of our knowledge, this is the first work to propose a spatial data structure for facilitating personalized indexing of various types of dynamically changing road conditions that could be associated with a given spatial location.

Incidentally, there have been several efforts towards sensing road conditions. For example, the Nericell system [9] uses smartphone sensors (e.g., accelerometer) to sense road conditions such as potholes. In the same vein, the research proposals in [6] and [14] use mobile sensors towards automated detection of speed breakers and anomalies on roads. Furthermore, crowdsourcing has been used as a means of involving residents by systems such as the FixMyStreet platform [1] and the Ushahidi platform [2]. The FixMyStreet platform encourages users to report road conditions (e.g., broken pavements, potholes), while the Ushahidi platform enables urban planners to take feedback directly from users. Thus, existing systems focus on how to collect the road condition data e.g., by means of automated sensors or via human involvement. However, they are not capable of facilitating users in performing quick and personalized retrieval of road condition data. Consequently, they cannot be effectively used for applications such as aiding the trip planning of an individual user based on her preferences concerning various types of road conditions.

III. System Architecture of RoadEye

This section discusses the system architecture of RoadEye. As depicted in Figure 3, the system comprises two key components: (a) an Event Reporting System (ERS) and (b) an Event Indexing and Analytics (EIA) engine. Additionally, the Communication System acts as an interface for facilitating communication between ERS and EIA.

ERS is responsible for data collection and pre-processing. Notably, data collection can be performed in different ways such as by deploying sensors, via crowd-sensing and so on. However, this work is agnostic to the data collection approach being used. Thus, it can be effectively used in conjunction with any existing data collection approach.

On the other hand, EIA consists of the following three components:

- **Event Pre-processing:** This component performs a completeness check on the event reports with the objective of pruning away the incomplete reports. For example, if a user reports a pothole, but fails to mention the location of the pothole, his event report is incomplete and therefore not of any practical use; hence such event reports should be deleted.
- **Event indexing:** This component is responsible for spatially indexing various event types associated with road conditions. In particular, it uses our proposed ψR-tree spatial index, whose description we defer to the next section.
- **Personalized Visualization Engine:** This component helps in providing personalized information about road conditions to the users. It can also be used for generating context-aware (e.g., spatial or temporal) alerts as depicted in Figure 1.

IV. ψR-tree: A Spatial Index for Quick and Personalized Retrieval of User-specified Road Conditions

This section presents the ψR-tree,which is a novel R-tree-based spatial index augmented with linked lists for facilitating quick and personalized retrieval of user-specified road conditions.

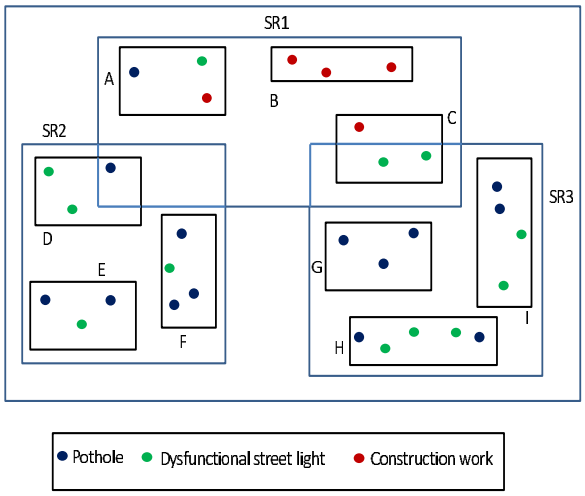


Fig. 4: An example for the distribution of event types in space

Non-leaf nodes of the ψR-tree are of the form (ptr, mbr, LL), where ptr is a pointer to a child node in the ψR-tree, and mbr is the MBR (Minimum Bounding Rectangle), which covers all the MBRs in that child node. Here, LL is a linked list structure, each node of which contains entries of the form ($event_{tag}$, $num_{instances}$), where $event_{tag}$ represents the type of event such as potholes, public processions, dysfunctional street lights and so on. Here, $num_{instances}$ refers to the frequency of occurrence of a given event type within a specific MBR.

Furthermore, the nodes of LL are kept sorted in descending order of the frequency of occurrence of the event types. The rationale is that if a specific type of event occurs more often within the MBR of a given region, the probability of users issuing queries pertaining to that event type is also likely to increase. For example, if the MBR of a region contains a significantly high number of potholes and speed bumps, users may be more likely to issue personalized queries to retrieve information about road conditions pertaining only to these two event types. Thus, keeping LL sorted facilitates in reducing the cost of traversing LL for the 'popular' user queries.

Leaf nodes of the ψR-tree contain entries of the form (SR_{id}, loc, LL_L), where SR_{id} is a pointer to a specific spatial region SR in the spatial database. Here, loc refers to the location of SR. Thus, loc can be specified either by one set of (x,y) coordinates if SR is a point in the granularity of the space being considered, or two sets of (x,y) coordinates if SR is a rectangle in that space. Furthermore, LL_L is a linked list, whose structure is essentially the same as that of the structure of the linked lists that are associated with the non-leaf nodes of the ψR-tree.

Observe that at the non-leaf node levels of the ψR-tree, the linked lists only contain event types (which occur within the MBR of the region covering the respective non-leaf node) and the respective frequencies of occurrence of each of these event types. However, in case of the regions covered by the MBRs at the leaf-node levels of the ψR-tree, much more detailed information about the road conditions can be retrieved by means of the pointer to the spatial database, where such detailed information can typically be stored. As a single instance, parameters pertaining to the average breadths and depths of the potholes or the average heights of speed bumps can also be accessed from the spatial database. Thus, the ψR-tree contains information about road conditions at finer granularity for the regions that are covered by the MBRs at the leaf-node levels than those at the non-leaf node levels.

Notice how the ψR-tree provides users with the flexibility to access both fine-grained as well as coarse-grained information about road conditions depending upon user preferences. The trade-off here is that accessing fine-grained information about road conditions (e.g., average height of speed bumps in a given region) would require spatial database accesses (i.e., higher disk I/O costs), thereby resulting in higher data access costs. On the other hand, queries for accessing coarse-grained information about road conditions entail significantly lower data access costs because such queries can be answered directly via the ψR-tree index without necessitating the need to perform costly spatial database accesses.

Figure 4 depicts an illustrative example for the distribution of event types in space, each event type being represented by a different colour. Observe how the universe is divided into three rectangular spatial regions $\{SR1, SR2, SR3\}$. $SR1$, $SR2$ and $SR3$ are further divided into the spatial regions $\{A, B, C\}$, $\{D, E, F\}$ and $\{G, H, I\}$ respectively. For simplicity, Figure 4 considers only three event types, namely *pothole*, *dysfunctional street light* and *construction work*.

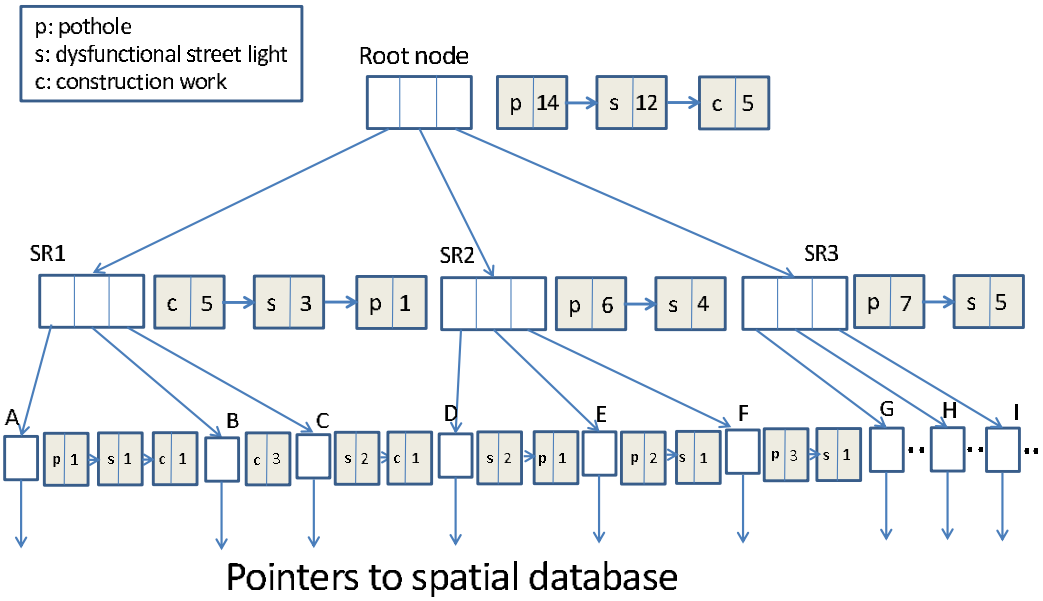


Fig. 5: Illustrative example for the proposed ψR-tree index structure

Fig. 6: Snapshot of the part of the United States Census Bureau dataset used in our experiments

Figure 5 depicts the structure of the ψR-tree index corresponding to the distribution of event types in space shown in Figure 4. As depicted in Figure 5, p, s, c stand for *pothole*, *dysfunctional street light*, *construction work* respectively. The linked list entries of the root node indicate that the region covered by the universe contains 14 potholes, 12 dysfunctional street lights and 5 instances of construction work. Furthermore, observe the linked lists corresponding to the different nodes of the ψR-tree. The significance of the entries in these linked lists is essentially the same as that of our explanation for the root node. As a single instance, the linked list entries of the non-leaf node $SR1$ contains 1 pothole, 3 dysfunctional street lights and 5 instances of construction work. For the sake of convenience, we have used this notation throughout the figure.

Observe how the entries in the linked lists are kept sorted in descending order of occurrence. Notably, for each type of event, the way in which the frequency of occurrence can be quantified is pre-defined based on the application. To provide some intuition, we could quantify the frequency of occurrence of construction work by measuring the number of different construction works that are currently on-going in a given area. Moreover, we could derive the frequency of occurrence by using more sophisticated approaches involving formulae to compute the frequency based on some weighted average

function of the relative sizes of each of these construction areas. In a similar vein, we could derive the frequency for a water-logging event based on formulae that consider the size of the water-logged area. However, as such, the quantification of the frequency of occurrence of any given type of event is outside of the scope of this work.

The ψR-tree can be populated based on information obtained from Geographical Information Systems (GIS) or provided by municipal or transportation-related government agencies. Creation of the ψR-tree from scratch uses the standard R-tree insertion algorithm, the only difference being the handling of the linked lists. Insertion of a new event requires a top-down traversal of the ψR-tree as follows. At each level of the ψR-tree and for each MBR (of the ψR-tree) that intersects with the spatial window of the event type, if a given event type is not present in a given linked list, it is added as a new node to the linked list at the appropriate position to keep the linked list sorted. Moreover, the corresponding frequency of occurrence is also stored.

Notably, the frequency of occurrence for a new event type need not necessarily be one e.g., five dysfunctional street lights could have been discovered at the same time in a given region. In the case that a given event type is already present in a given linked list, the frequency of occurrence is simply incremented by the number of event instances of that event. For example, if the pothole event type already exists in a given linked list, and five more potholes get discovered, the frequency of occurrence for this event type would simply be incremented by five.

Deletion of obsolete events is required to keep the ψR-tree updated. Intuitively, we can understand that the algorithm for deleting events from the ψR-tree also proceeds in a similar vein, except that the number of deleted event instances are subtracted from the frequencies stored at the respective linked lists as opposed to being added. For example, if it is discovered that a specific number of potholes have been repaired in a given region, the frequency for the corresponding linked list entries would need to be decremented accordingly. Furthermore, if the frequency for a given node of a linked list becomes zero as a result of the decrement, that node simply needs to be deleted from the linked list.

V. Performance Evaluation

This section reports our performance evaluation. The proposed ψR-tree indexing structure has been implemented in Java. All our experiments have been performed on a 64-bit machine with 8 GB memory and 2 CPU cores.

In our ψR-tree index implementation, each rectangle was represented using two (x,y) coordinates; hence 16 bytes were used to represent each of the input rectangles with 4 bytes for each child pointer. We used a branching factor of 64, thereby resulting in (64*20) i.e., 1280 bytes of space consumption. Furthermore, the linked list structure at each R-tree node requires 4 bytes for the event type, another 4 bytes for the frequency of event occurrence and 4 bytes for each pointer. Given that we consider the maximum possible number of event types to be 100, the space consumption arising from the linked list structure at each R-tree node is (100 * 12) i.e., 1200 bytes. Additionally, 4 bytes are needed for the pointer connecting the R-tree node to the linked list structure. Therefore, the maximum space consumption of each augmented R-tree node is (1280 + 1200 + 4) i.e., $\approx$ 2.4 KB.

We selected the disk page size to be 4 KB and each augmented R-tree node is assumed to fit in a disk page. Hence, for our experiments, one disk I/O is counted as one disk page access or equivalently one augmented R-tree node access. Observe that each disk page has more than 1.5 KB of free space, which can be used to accommodate future updates to the ψR-tree without necessitating expensive node-splits.

We evaluate the performance of the ψR-tree on a part of the publicly available road network dataset of the United States Census Bureau[1]. The part of the dataset used in our experiments comprises 1 million rectangular bounding boxes of road segments. In this dataset, the road network is represented by shape files, and these rectangular bounding boxes were extracted from those shape files. A snapshot of the dataset is shown in Figure 6.

We divide the universe into a grid of $10X10$ equal-sized cells. We sort the cells in descending order of the number of input data rectangles. Given the highly skewed spatial dataset used in our experiments, there would be a relatively few 'hot' cells containing a disproportionately large number of input data rectangles, while the other cells would each contain a relatively low number of input data rectangles.

Event types are associated with input data rectangles as follows. We use a highly skewed Zipf distribution with a zipf factor of 0.7 to ensure that some event types have a significantly higher probability of occurrence than the other event types. Given an input rectangle, a random number Rnd is generated between 0.0 and 1.0 for each event type. If Rnd is less than the probability (generated using the Zipf distribution) for that event type, the event type is considered to have occurred within that input rectangle; otherwise the event type is deemed not to have occurred within that input rectangle.

As reference, for the purpose of meaningful performance comparison, we use an approach designated as **MR (Multiple R-trees)**. MR creates and maintains separate R-trees, one for each event type. Thus, for a total of N_E event types, MR would use N_E separate R-trees. For example, all the pothole-related events would be indexed by an R-tree, while all the events related to dysfunctional street lights would be indexed by a different R-tree. Thus, for a user query containing multiple event types, MR entails the traversal of multiple R-trees. Observe that this is in contrast with our proposed ψR-tree approach, which requires the traversal of only one R-tree-based index i.e., the ψR-tree. This is made possible because the ψR-tree is capable of indexing multiple road condition events in a single R-tree structure due to its novel linked list augmentation at each R-tree node.

Our performance metrics are **average response time (ART)**

[1] http://www.census.gov/geo/maps-data/data/tiger-line.html

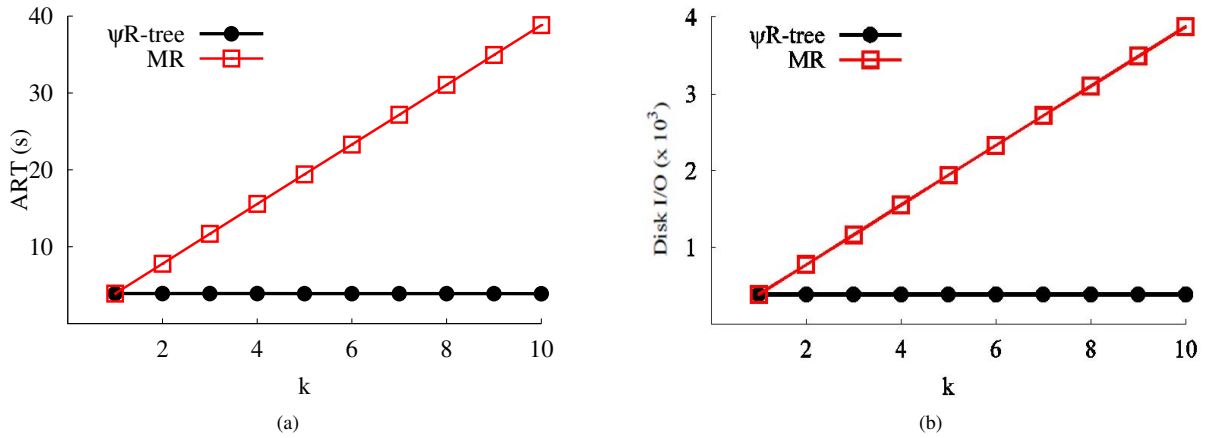


Fig. 7: Effect of variations in number k of user-specified event types in a a given query

Parameter	Default	Variations
No. of input rectangles	10^6	
No. of possible event types (N_E)	100	
Query scaling factor (Q_{SF})	1	1.5, 2, 2.5, 3, 3.5, 4
No. of user-queried event types (k)	6	1,2,3,4,5,7,8,9,10
Disk Page Size	4 KB	
ψR-tree branching factor	64	

TABLE I: Performance study parameters

and **disk I/Os** incurred by the queries in our experiments. Here, $ART = (1/N_Q)\sum_{i=1}^{N_Q}(T_{ci} - T_{ai})$, where T_{ai} represents the time of query issuing and T_{ci} is the time of completion for the i^{th} query. N_Q is the total number of queries. The disk I/O metric measures the corresponding number of disk I/Os incurred for traversing the R-tree-based index structure(s) during the course of each experiment. Hence, in case of the ψR-tree approach, the disk I/O metric measures the disk I/Os incurred for traversing the ψR-tree. On the other hand, for the MR approach, the disk I/O metric measures the total number of disk I/Os incurred for traversing each of the R-trees relevant to the event types specified in the query.

Table I provides a summary of the parameters used in our performance evaluation. N_E represents the total number of possible event types. In real-world scenarios, the value of N_E would typically not exceed 100; hence we set N_E = 100 for our experiments. A given query is characterized by the area $Area$ of its spatial window as well as the user-queried event types in it. We define $Area$ as a percentage of the total area of the universe under consideration. Based on the results of our preliminary experiments, we set $Area$ to 5% of the universal space. Furthermore, we use a scaling factor, designated as Q_{SF} to study the effect of varying $Area$. When Q_{SF} = b, both the length and width of the rectangular spatial window would be multiplied by b. For example, suppose $Area$ = 40 and b = 1.25, the area of the scaled rectangle would be $40 * 1.25^2$ i.e., 62.5. In Table I, k is the number of user-specified event types in a given query.

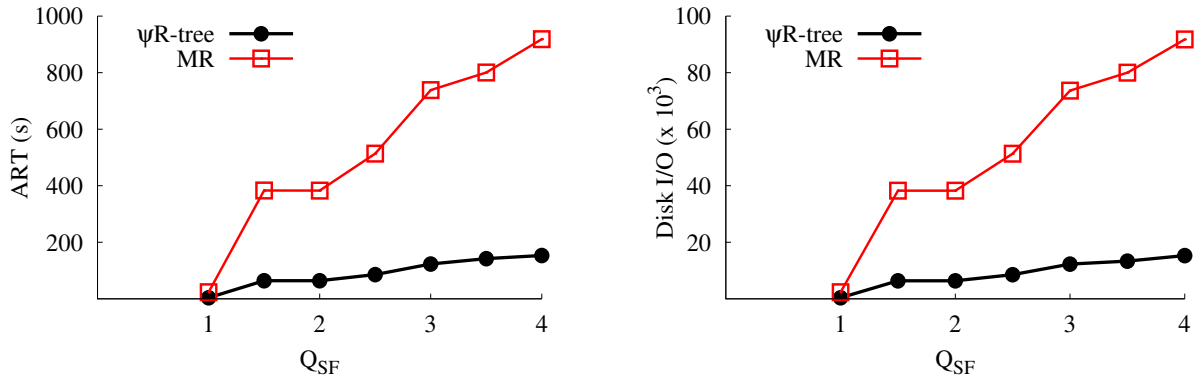
Notably, a significantly larger number of queries are likely to come from the relatively 'hot' cells. Hence, we use a highly skewed Zipf distribution with zipf factor of 0.7 to determine the probability of a query originating from a particular cell. A given query is generated as follows. (a) A cell is selected based on the above Zipf distribution. (b) A point is selected randomly from that cell. (c) A rectangle of size $Area$ is constructed around the selected point. (d) Finally, any k event types are randomly selected from the total number N_E of 100 possible event types.

We ran each experiment at least five times and the results presented here represent the average of those runs. The confidence interval of our experimental results is 95%.

A. *Effect of variations in k*

Figure 7 depicts the results for the effect of variations in the number k of queried event types. Observe that with increase in k, the reference MR approach exhibits a significant increase in both ART and disk I/Os. This occurs because as k increases, MR has to traverse an increased number of R-trees, thereby incurring higher number of disk I/Os. On the other hand, as Figure 7 suggests, ART and disk I/Os remain comparable for the ψR-tree with increase in k. This is because the ψR-tree is capable of indexing multiple event types in a single R-tree structure due to its novel augmentation of the R-tree nodes by incorporating linked lists. Thus, in case of the ψR-tree, the disk I/O cost remains comparable, irrespective of the value of k. Furthermore, this also explains the increase in performance gap between the two approaches with increase in k.

Interestingly, a detailed examination of the experimental logs revealed that ART increases (albeit slightly) with increase in k in case of ψR-tree. This occurs because with increase in k, the time taken to traverse the linked lists increases. However, this increase in ART is negligible because the cost of traversing the linked lists is dwarfed by the predominant disk I/O costs of ψR-tree node retrievals. Furthermore, observe that ART and disk I/O follow comparable trends for both the approaches essentially due to the reason explained above.



(a) (b)

Fig. 8: Effect of variations in query scaling factor Q_{SF}

B. *Effect of variations in* Q_{SF}

Figure 8 depicts the results for the effect of variations in the scaling factor Q_{SF} for a given query. Recall that we use Q_{SF} to vary the area of the query window. As Q_{SF} increases, both the approaches exhibit increase in both ART and disk I/Os. This occurs because increased query window area results in the query intersecting a relatively larger number of index nodes. However, the ψR-tree shows relatively lower increase in both ART and disk I/Os with increase in Q_{SF}. Furthermore, the performance gap between the two approaches exhibits an increasing trend with increase in Q_{SF}. The rationale for this performance gap can essentially be explained in the same manner as for the results in Figure 7.

VI. Conclusion

In this work, we have addressed the problem of keeping users informed in a *timely* and *personalized* manner about dynamic road conditions. We have presented RoadEye, which is a system for *personalized* retrieval of dynamic road conditions. In particular, RoadEye incorporates our proposed ψR-tree, which is a novel R-tree-based index augmented with linked lists for facilitating quick and personalized retrieval of user-queried road conditions. Our performance study indicates the overall effectiveness of the ψR-tree in retrieving dynamic road conditions with reduced query response times and disk I/Os.

In the near future, we intend to study the effect of incorporating event type hierarchies into the ψR-tree index. For example, an event type such as 'obstacles on the road' would encompass events such as fallen trees on the road, snow on the roads and so on, thereby forming a hierarchy of event types. Moreover, we intend to examine the effect of skews in the event type distributions in space as well as the use of hash tables to augment our proposed R-tree-based data structure. Finally, we plan to investigate the feasibility of extending the ψR-tree to go beyond road conditions so that it can be relevant to other diverse and important aspects of smart city management.

References

[1] Fixmystreet platform. http://www.fixmystreet.com.
[2] Ushahidi platform. http://blog.ushahidi.com/2012/06/05/ushahidibeijing.
[3] Tamas Abraham and John F. Roddick. Survey of spatio-temporal databases. In Proc. of Geoinformatica, 1999.
[4] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In Proc. of ACM SIGMOD, 1990.
[5] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In Proc. of ACM SIGMOD, 1984.
[6] Mohit Jain and Ajeet Pal Singh. Speed breaker early warning system. In Proc. of USENIX/ACM Workshop on Networked System for Developing Regions, 2012.
[7] Nick Koudas, Christos Faloutsos, and Ibrahim Kamel. In Proc. of EDBT, 1996.
[8] Dongseop Kwon, Sangjun Lee, and Sukho Lee. Indexing the current positions of moving objects using the lazy update R-tree. In Proc. of MDM, 2002.
[9] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In Proc. of ACM Conference on Embedded Network Sensor Systems, 2008.
[10] Anirban Mondal, Masaru Kitsuregawa, Beng Chin Ooi, and Kian Lee Tan. R-tree-based data migration and self-tuning strategies in shared-nothing spatial databases. In Proc. of ACM GIS, 2001.
[11] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches in query processing for moving object trajectories. In Proc. of VLDB, 2000.
[12] Bernd Schnitzer and Scott T. Leutenegger. Master-client R-trees: A new parallel R-tree architecture. In Proc. of SSDBM, 1998.
[13] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In Proc. of VLDB, 1997.
[14] Yu-chin Tai, Cheng-wei Chan, and Jane Yung-jen Hsu. Automatic road anomaly detection using smart mobile device. In Proc. of Conference on Technologies and Applications of Artificial Intelligence, 2010.
[15] Yufei Tao and Dimitris Papadias. MV3R-tree: A spatio-temporal access method for timestamp and interval queries. In Proc. of VLDB, 2001.
[16] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. In Proc. of ACM SIGMOD, 2000.