Indexing techniques for Geospatial Searching : A survey

Amruta Joshi, Prof. U. M. Patil

Abstract— These Given a geographic query that is composed of query keywords and a location, a geographic search engine retrieves documents that are the most textually and spatially relevant to the query keywords and the location, respectively, and ranks the retrieved documents according to their joint textual and spatial relevance to the query. In this survey paper, the efficient index, called IR-tree, that together with a top-k document search algorithm is studied along with similar techniques like KR* Trees, and other hybrid indices

Index Terms— GeoSpatial searching,Spatial Keywords(SK),Document frequency, Term frequency Minimum

1 INTRODUCTION

An index based on what the object spatial locations is desirable, but classical one dimensional database indexing structures are not appropriate to multi-dimensional spatial searching Structures based on exact matching of values, such as hash tables, are not useful because a range search is required Structures using one dimensional ordering of key values, such as B-trees do not work because the search space is multi-dimensional. A number of structures have been proposed for handling multi-dimensional point data [11].
Spatial data objects often cover areas in multi-dimensional spaces and are not well represented by point locations For example, map objects like counties,census tracts etc occupy regions of non-zero size two dimensions Techniques such as relevance feedback, thesaural expansion, and pivoting all provide better quality responses to queries when tested in standard evaluation frameworks

2 INDEXING MECHANISMS

Currently, two types of approaches are used by existing geographic search engines, namely
1) separate index for spatial and text attributes
2) hybrid index that combine spatial and text attributes.

2.1 Separate Index for Spatial and Text Attributes

In this approach, separate index structures are built for spatial data and text data. Based on two indexes, a search generally follows a three step process.
. Step 1: retrieving textually relevant documents with respect to query keywords via a conventional textual index.
. Step 2: filtering out the documents obtained from Step 1 that are not covered by the query spatial scope.

- *Amruta Joshi is currently pursuing masters degree program in compter engineering in RCPIT, Shirpur, North Maharashtra University, Maharashtra, India.*
- *Prof. U. M. Patil is currently working in RCPIT, Shirpur,Maharashtra, India.*

. Step 3: ranking the documents from Step 2 based on the joint textual and spatial relevances in order to return the ranked results to the user.

The choice of index structure for spatial data can be grid, quadtree, or R*-tree. One commonly used structure is R*-tree [19], and the choice of other indices are also possible. For text, [18] proposed an inverted file index. The inverted file index stores for each keyword, a sorted list of object ids in which the keyword appears, its score, and frequency.
Using this approach, *SK* queries can be answered in two ways. First, a set of candidate object ids that satisfy the spatial part of the query are retrieved using the spatial index. The object ids are sorted and for each retrieved object id, the textual keywords of the query are looked up in its corresponding inverted list index. Finally, all the object ids that satisfy the query are collected, ranked and presented as sorted results to the user. The second approach is to first filter object ids based on the query keywords. Inverted index list for each query keyword are looked up, and a set of object ids that are present in the intersection of the lists are passed to the next stage for, spatial filtering. Finally, the scores for each object are computed by combining the ranking of textual and spatial parts. The performance of both approaches depends on the selectivity of the objects satisfying the text or spatial part of a query. If the number of objects in the spatial region of the query is small, it is better to do the spatial filtering first and vice versa. In [11], the choice for the spatial index is a grid and inverted file index for textual keywords.
In [3], the authors propose a number of improved techniques to the above basic approaches. First, they suggest storing spatial data in the disk by following Hilbert curve ordering [6]. This ordering maintains the spatial closeness of objects thereby speeding up the disk access operations in retrieving the spatial data. Secondly, the objects in the inverted index list are assigned ids according to Hilbert ordering and sorted based on these ids. A grid-based structure is built in memory to store the ids of the spatial data in each tile of the grid. When a query is issued by the user, the relevant tiles of the grid that overlap the spatial region of the query are retrieved. The object ids contained in the tiles are sorted and looked up against the inverted indices.
Advantages and Limitations: The main advantage of the above strategies is the ease of maintaining two separate indi-

ces. However, the main performance bottle -neck lies in the number of candidate objects generated during the filtering stage. If spatial filtering is done first, many objects may lie within a query's spatial extent, but very few of them are relevant to the query keywords. This increases the disk access costs by generating a large number of candidate objects. The subsequent stage of keyword filtering becomes expensive. The same is true, if keyword filtering is done first. Moreover, the above strategies assume a memory resident spatial index which is not reasonable for large GIR databases. By A. Ntoulas and J. Cho, this issue is discussed by proposing to reduce the granularity of spatial index, so that it fits in main memory. However, if the grid is too coarse, it loses its pruning capabilities. We build a disk resident spatial index.

2.2 Hybrid Indices

Hybrid indexing techniques combine the spatial and inverted file indices. A. Ntoulas and J. Cho, the inverted list was modified as the following. The list for each keyword was augmented with the space in which the objects contained in the list appear. For instance, if $w1$ is the keyword, its list was augmented as: $w1 = \{r1(o1, o2, ...), r2(o2, ...), ...\}$ where $r1$, $r2$, .. are bounding rectangles in space. When a query is issued, the corresponding keyword lists are loaded, and objects are filtered using the associated spatial index. This strategy still requires scanning the entire list. The closest work to ours is the hybrid indexing structures proposed in [22]. The first hybrid data structure shown in Figure 2 is called *Inverted File-R*-tree*.
The second data structure proposed in [2] is called *R*- tree-Inverted File*.

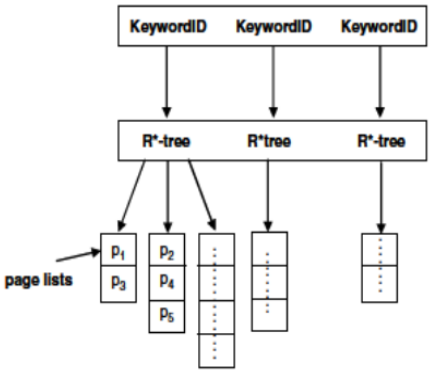

Fig. 2.1 Structure of first inverted file then R*-tree

The name of a location and every word of a document are combined as a new word. Then, an inverted file based on those new words is created to support geographic searches. However, this approach simply treats locations as texts and cannot deal with various spatial relevance computations. On the other hand, two hybrid indexes are proposed, namely, 1) an inverted file on top of Rtrees referred to as HybridI, and 2) an R-tree on top of inverted files referred to as HybridR. Thus, a search upon HybridI first locates a collection of documents based on search keywords and then based on locations. The
search strategy is reversed for HybridR. However, these hybrid indexes do not integrate the textual filtering and spatial filtering seamlessly.

Along the same line, IR2-tree [8] builds an R-tree and uses signature files (rather than a set of words) to record the document words associated with nodes in the index.
Signature files reduce the storage overhead and R-tree can quickly determine the documents spatially covered by a query spatial scope. However, signature file can only determine whether a given document contains query keywords but fail to order them based on the textual relevance.

Advantages and Limitations:

The first approach proposed in "Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems" is highly insensitive to *SK* queries with the *AND* semantics. This approach does not take advantage of the association of keywords in space. Hence, when query contains keywords that are closely correlated in space, this approach suffers from paying extra disk costs accessing different R*- trees and high overhead in the subsequent merging process. In the second approach proposed in [9], the leaf nodes point to inverted index lists that are usually small. This is because any
ne entire data
keywords that
This approach
eyword filter-
main disad-
enerates many

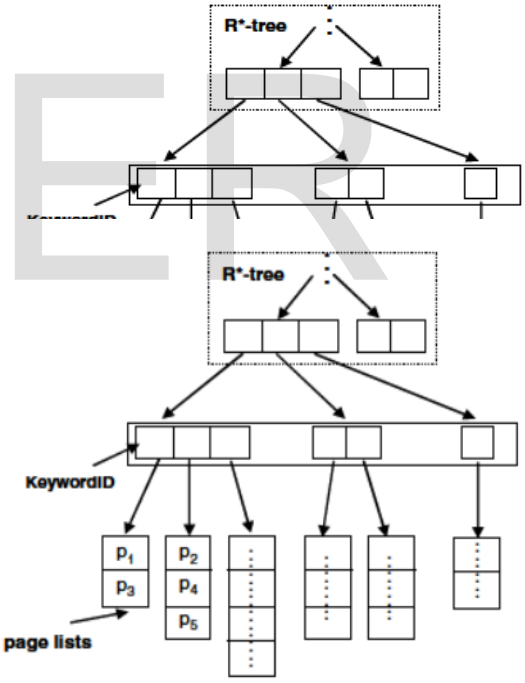

Fig. 2.2 Structure of first R*tree then inverted file

3. RELATED WORK

Implicit Locations :

There are several possible solutions to the implicit location problem. One method is to use query expansion based on pseudofeedback [4]. The idea is to find the most relevant locations to the original query location from the returned documents, and then use those locations to compose a new query. The shortcoming of the pseudo-feedback approach is that it

depends on the documents so much that many irrelevant locations might be added to the new query. Another method is to expand the query based on gazetteers. The locations that are covered by the query location will be used for expansion. Unfortunately, we may come up with a very long
query after such an expansion because too many children locations will be added. Moreover, such a long expansion will greatly affect the retrieval speed. Geo-index structures like gridbased index and R*-tree can solve the implicit location problem
too. In this paper, we assume the query location is input by text which is common for current search engines.in the approach, explicit locations and implicit locations are indexed together and different geo-confidence scores are assigned to them. The advantage of this mechanism is that no query
expansion is necessary and implicit location information can be computed offline for fast retrieval.In [6], the authors concluded that the most common geographical relationship used in queries is "in". Actually, if no spatial relationship exists in the query, we can safely assume the relationship is "in". In this paper, we adopt two types of geo-indexes: one is called focus-index, which utilizes the inverted index to store all the explicit, and implicit locations of documents (see Figure 1); the other is called grid-index, which divides the surface of the Earth into 1000 × 2000 grids. All the documents will be indexed by these grids according to their geo focuses. The reason for adopting grid-index is that some topics in GeoCLEF can't be solved by only focus-index due to the spatial relationship other than "in" (like "near"). For focus-index, the matched docID list can be retrieved by looking up the locationID in the inverted index. For grid-index, we can get the docID list by looking up the grids that the query location covers. We first retrieve two lists of documents relevant to the textual terms and the geographical terms respectively, and then merge them to get the final results. A combined ranking function

$$R_{combined} = R_{text} \times \alpha \; R_{geo} \times (1-\alpha),$$

where Rtext is the textual relevance score and Rgeo is the geo-confidence score, is computed and used to re-rank the results. Experiments show that
textual relevance scores should be weighted higher than georelevance scores. In these projects, the main objective is to address the extraction of geographic references found in the text by using ontologies, gazetteers, thesaurus, etc., and convert them to coordinates for retrieving DL contents using geography.

In the context of geographic search engines, there are numerous academic projects. Most of them can be broadly classified under 1) work that focused on extraction of geographic references from documents and/or 2) efficient query processing. We will briefly describe a few of these. In GeoSearch System [10], the geographic scope of Web pages are extracted by analyzing the geographic references in text as well as the geographic location where the Web sites are registered. In [15], the focus is on improving the extraction techniques. In particular, after the relevant geographic references are extracted, ambiguities such as multiple place name references and alternate place names are resolved using techniques such as geo-matching and geo-propagation.

In the context of query processing for GIR, indexing techniques for processing text and geographic data are the main focus. In [7], a simple inverted index structure for text and grid file for geographic data are used. They propose a hybrid index structure in which each keyword is combined with different partitions of space. In effect what they are proposing is similar to [9].

In a recent work [3], the authors propose to maintain individual indices for spatial and textual data. They propose various approa-ches to retrieve data from each index before the final merging of results. The spatial objects indexed in their applications are complex footprints that are extended regions in space. They approximate them by using MBRs and use memory-resident spatial index. Their approach does not scale well with increasing size of the dataset. To alleviate the problem, they propose to compress the MBRs, but the attempt generates large candidate set that
needs to be fetched from the disk, with a high rate of false positives. This will become a major performance bottleneck for large scale GIR applications. In our work, we use disk-resident spatial index for GIR applications. Our data structure performs significantly better than their approach with respect to two aspects:

1) first it reduces the number of disk accesses in identifying the candidate objects and as a consequence
2) it reduces the overhead in merging the candidate objects.

In "Hybrid Index Structures for Location-based Web Search"another very related work, the authors proposed a hybrid index by combining the spatial and inverted list structures. Their approaches either use multiple R*-trees to answer queries or generates more candidates for further filtering.

First of all, a keyword-based search may retrieve a large number of textually relevant documents that are outside the spatial scope. Although it is possible to reorder Steps 1 and 2 based on their selectivities, performance improvement is rather limited if the selectivities in Steps 1 and 2 are both high. Besides, the ranking process is not incremental, i.e., it has to sort all of the candidate documents based on the joint textual and spatial relevances
in Step 3 in order to find the top-k documents. As the total number of candidate documents is usually much larger than k, document ranking becomes very expensive. Further, these three steps are performed sequentially, prolonging the
processing time and requiring a large memory storage to buffer intermediate results between steps.

To improve the search efficiency, Approach II combines the spatial locations and textual contents of documents together and builds one index on them.

3.1 Inverted File and R*-tree Double Index

In this structure, web pages are indexed separately twice, once by R*-tree and once by inverted files. All MBRs are indexed by an R*-tree. The difference from conv-entional R*-tree is that each leaf node of the MBR tree points to a page list whose scope includes this MBR. Inverted files are the same to conventional search engines. Thus we have two kinds of page lists whose entry is either an MBR or a keyword. A location-based

web search comprises non-spatial keywords and query region and/or specified spatial query types. Non-spatial query keywords are retrieved similar to conventional inverted files, while query region and spatial query type are passed to the R*-tree. The final results are the merge of page lists from two indexes.

The storage in disk comprises the two kinds of page lists and the R*-tree. The storage of page lists depends on the length of each list, whose unit is the identifier of a page Assuming the length of the list whose

entry is keyword k is $PK(k)$ and the length of the list whose entry is MBR m is $PM(m)$

So, the main cost of storage in disk is caused by two kinds of page lists above. For the storage of the identifier of pages is about a fixed value, the storage is mainly determined by the total length of all page

lists. The time of loading page lists is determined mainly by the number and total length of lists. The merge processing depends on the total length of these page lists.

3.2 First Inverted File Then R*-tree

For each page list with an entry is a keyword in the first hybrid structure, these pages in the list are assigned to different MBRs according to their geographical scope. R*-tree is built on these MBRs, to get a set of page lists whose entry is determined by a pair of a keyword and an MBR. A pair of a keyword and an MBR is named a geo-keyword if there is a page which includes the keyword and whose scope includes the MBR. The scale of R*-trees is smaller. The cost of storage in disk is mainly caused by the total length of page lists whose entry is a geo-keyword. Assume that the number of geo-keywords for a query Q of m keywords and n MBRs is $g(Q)$. The online computation includes: (1) first to retrieve the m query keywords; (2) to search in the corresponding

R*-trees whose number is m and the average leaf node is M , and to find some MBRs and their corresponding page lists, the number of lists got from m R*-trees is $g(Q)$; (3) to merge these $g(Q)$ page lists.

The retrieval for m keywords is implemented by a hashing function, and the time is ignored. So, Besides the retrieval of m R*-trees, there are also two main factors for online search. One factor is caused by the total length of the page lists whose entry is a geo-keyword, the number of lists is $g(Q)$. The other factor is the time to read these $g(Q)$ page lists from disk. The main storage in disk includes the page lists whose entry is a geo keyword and the R*-tree. The main cost of storage in disk is caused by the total length of page lists whose entry is a geo-keyword.

D. Felipe, V. Hristidis, and N. Rishe, states in the paper "Keyword Search on Spatial Databases,"IR^2-tree builds an R-tree and uses signature files (rather than a set of words) to record the document words associated with nodes in the index. Signature files is used to reduce the storage overhead and R-tree can quickly determine the documents spatially covered by a query spatial scope. However, signature file can only determine whether a given document contains query keywords but can't order them based on the textual relevance. [8]

TABLE 1:
LITERATURE SURVEY OF INDEXING DATA STRUCTURES

Author	Indexing data structure	Publication	Year
Zhisheng Li, Ken C.K. Lee,	IR-Trees	IR-Tree: An Efficient Index for Geographic Document Search	2011
I.D. Felipe, V. Hristidis, and N. Rishe	IR2-tree [8]	Keyword Search on Spatial Databases	2008
Ramaswamy Hariharan, Bijit Hore	KR* trees	Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems	2007
Y.-Y. Chen, T. Suel, and A. Markowetz	Quad-tree	Efficient Query Processing in Geographic Web Search Engines	2006
K.S. McCurley,	Grid index	Geospatial Mapping and Navigation of the Web	2001
A. Guttman	R- Trees	"R-Trees: A Dynamic Index Structure for Spatial Searching	1984

3.3 KR*-Tree:

Zhisheng Li, Chong Wang,et.al, proposed a new indexing strategy called *KR*-tree*, which is an acronym for Keyword-R*-tree. We measure the effectiveness of indexing strategies in answering *SK* queries with respect to the following criteria:

- Pruning text and space.
- Handling queries with multiple keywords.

First, the indexing methods previously proposed use the pruning power of space and text, either separately or one followed by the other. As a consequence, *SK* queries are answered in a two-step filtering process, space followed by text or vice-versa. In *KR*-tree*, we exploit the pruning power of both space and text simul-taneously, thus merging the two steps into one. Secondly, in previous methods the keywords are maintained separately. Hence queries are answered by the intersection of object ids from the inverted index file or R*-trees of query keywords. In *KR*-tree*, we capture the joint distribution of key-

words and hence the object ids containing the query keywords are directly obtained without merging any lists. These characteristics greatly enhance the performance of *KR*-tree* in answering *SK* queries. At the outset, *KR*-tree* is similar to *R*-tree-Inverted File* data structure, but with the following modifycations.

- All internal and leaf nodes of *KR*-tree* are augmented with a set of distinct keywords that appear in the space covered by the nodes. Thus, many keywords appear in the upper level nodes of the tree and smaller number of keywords appear in the lower level nodes of the tree.
- Since the number of keywords that appear in each node varies, we do not store the keywords in the node. We construct a special list called *KR*-tree List* that stores the keywords appearing in the nodes[9].

Limitations of KR*-tree :

KR*-tree and IR2-Tree are not efficient due to separation of document search and document ranking. After the document search step, a large number of candidate documents are usually retrieved but only k of them are returned after document ranking. Consequently, the evaluation of those nonresult candidates is a waste. Although KR*-tree, IR^2-Tree and IR-tree are built on top of R-tree, they are very different in terms of structures, functionalities, and extensibility to searches with various relevance requirements[9].

TABLE 2.
OPTIMIZING TECHNIQUES FOR REDUCING QUERY COST

Author	Publication	Year	Remarks
Alexandros Ntoulas_ Junghoo Cho†	Pruning Policies for Two-Tiered Inverted Index with Correctness Guarantee	SIGIR’07 .	It is based on two tire indexing strategy and pruning at two levels viz. text pruning and document pruning.
S. B¨uttcher and C. L. A. Clarke	A document-centric approach to static index pruning in text retrieval systems	CIKM, 2006.	It uses static index pruning along with the document pruning.
V. N. Anh and A. Moffat.,	. Pruning strategies for mixed-mode querying.	CIKM, 2006.	[illegible]
D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. Maarek, and A. Soffer.	Static index pruning for information retrieval systems	SIGIR, 2001.	[illegible]
V. N. Anh, O. de Kretser, and A. Moffat	Vector-space ranking with effective early termination.	SIGIR, 2001.	[illegible]
S. Chaudhuri and L. Gravano	Optimizing queries over multimedia repositories.	. In SIG-SIG-MOD, 1996	It has the specific repository for multimedia updates to reduce the updating query costs in frequently changing

			enviro

3.4 IR-trees:

IR-tree, proposed by Zhisheng Li, et.al. is an efficient index that provides the following required functions for geographic document search and ranking:

1) Spatial filtering: all the spatially irrelevant documents have to be filtered out as early as possible to shrink the search space;

2) Textual filtering: all the textually irrelevant documents have to be discarded as early as possible to cut down the search cost; and

3) relevance computation and ranking: since only the top-k documents are returned and k is expected to be much smaller than the total number of relevant documents, it is desirable to have an incremental search process that integrates the computation of the joint relevance and document ranking seamlessly so that the search process can stop as soon as identification of the top-k docu-ments. IR-tree is designed by taking into account the storage and access overheads since a document set is very large in terms of numbers of documents and their words.

IR-Tree Structure

In order to support efficient geographic document search, the IR-tree proposed by Zhisheng Li, et.al. clusters a set of documents into disjointed subsets of documents and abstracts them in various granularities. By doing so, it enables the pruning of irrelevant subsets. The effi-ciency of IR-tree depends on its pruning power, which, is highly related to the effectiveness of document clustering and the search algorithms. IR-tree clusters spatially close documents together and carries textual information in its nodes. These designs distinguish IR-tree from other hybrid indexes. IR-tree associates each leaf entry with an inverted file and associates adocument summary that provides textual information of docu-ments with each node so that the tf and idf values of the document words can be estimated at nodes without examining individual documents.

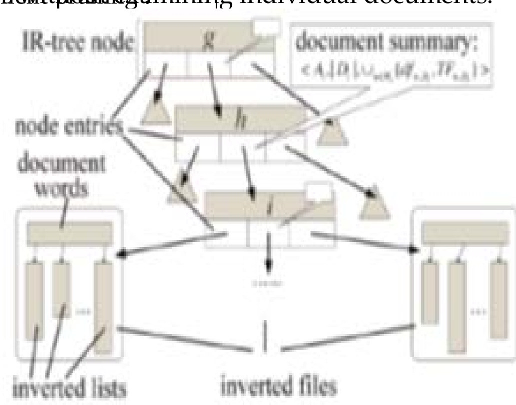

Fig. 3.3 Structure of IR-Tree [1]

Figure 3.3 depicts an IR-tree indexing structure. An inverted file consists of a list of words, with each corresponding to a

word w and pointing to a list of documents that contain w. Then, for each node i, a document summary about a set of documents Di indexed beneath i is captured as a three-element tuple: Each document d in a given document set D a set of words W_d, associated with a location L_d for each node i, a document summary about a set of documents D_i

- $<A_i, |D_i|; U_{w \in W_i}\{df_{wDi}, TF_{w,Di}\}>$
- A_i the minimal bounding box covering all of the locations L_d of documents d in D_i
- $W_i\{df_{wDi}, TF_{w,Di}\}$ -: each word w that appears in at least one document in D_i (i.e., W_i),

Next, ,$|D_i|$ refers to the cardinality
of the document set D_i. The third element is a set of$(df_{wDi}, TF_{w,Di})$ pairs. For each word w that appears in at least one document in D_i (i.e., W_i), df_{wDi} represents the number of documents in D_i that contain w and $TF_{w;Di}$ is the aggregated information about the tf values of w in Di. We investigate two different representations of $TF_{w;Di}$, Notice that the document summary of a non root node i is stored with i's parent node h. Then, given a query that reaches i's parent node h, it can decide whether i contains potential result documents (i.e., whether the examination of i is necessary)based on the document summary.

REFERENCES

[1]Zhisheng Li, Ken C.K. Lee, Member, IEEE, Baihua Zheng, Member, IEEE, Wang-Chien Lee, Member, IEEE, Dik Lun Lee, and Xufa Wang, IEEE, "IR-Tree: An Efficient Index for Geographic Document Search", Transactions on knowledge and data engineering, vol. 23, no. 4, April 2011

[2]moonbae song, hiroyuki kitagava , "managing frequent updates in R-trees for update-intensive applications", IEEE Trans. Knowledge and Data Eng. (TKDE), vol. 21, no. 11, pp. 45-55, Nov./Dec. 2009.

[3]A. Ntoulas and J. Cho, "Pruning Policies for Two-Tiered Inverted Index with Correctness Guarantee," Proc. ACM SIGIR '07, pp. 191-198, 2007.

[4]S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.-T. Lu,"Spatial Databases—Acco4mplishments and Research Needs," IEEE Trans. Knowledge and Data Eng. (TKDE), vol. 11, no. 1, pp. 45-55, Jan./Feb. 1999.

[5] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid Index Structures for Location-Based Web Search," Proc. 14th ACM Int'l Conf. Information and Knowledge Management (CIKM '05), pp. 155-162, 2005.

[6] I.D. Felipe, V. Hristidis, and N. Rishe, "Keyword Search on Spatial Databases," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08),pp. 656-665, 2008.

[7] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD '84, pp. 47-57, 1984.

[8] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems," Proc. 19th Int'l Conf. Scientific and Statistical Database Management (SSDBM '07), pp. 16-25, 2007.

[9] Ramaswamy Hariharan, Bijit Hore, Chen Li, Sharad Mehrotra, Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems, Proc. 19th Int'l Conf. Scientific and Statistical Database Management (SSDBM '07), pp. 16-25, 2008.

[10] Zhisheng Li1, Chong Wang2, Xing Xie2, Xufa Wang1, Wei-Ying Ma2,Indexing implicit locations for geographical information Retrieval, Proc. 19th Int'l Conf. Scientific and Statistical Database Management (SSDBM '07), pp. 16-25, 2009.

[11] Nieves R. Brisaboa ,Miguel R. Luaces , Ángeles S. Places ,Diego Seco "Exploiting Geographic References of Documents in a Geographical Information Retrieval System Using an Ontology-based Index"., Proc. Third Workshop Geographic Information Retrieval (GIR '06), 2006.