See-To-Retrieve: Efficient Processing of Spatio-Visual Keyword Queries

Chao Zhang Lidan Shou Ke Chen Gang Chen
College of Computer Science
Zhejiang University, China
{chaozhang,should,chenk,cg}@zju.edu.cn

ABSTRACT

The wide proliferation of powerful smart phones equipped with multiple sensors, 3D graphical engine, and 3G connection has nurtured the creation of a new spectrum of visual mobile applications. These applications require novel data retrieval techniques which we call What-You-Retrieve-Is-What-You-See (WYRIWYS). However, state-of-the-art spatial retrieval methods are mostly distance-based and thus inapplicable for supporting WYRIWYS. Motivated by this problem, we propose a novel query called *spatio-visual keyword* (SVK) query, to support retrieving spatial Web objects that are both visually conspicuous and semantically relevant to the user. To capture the visual features of spatial Web objects with extents, we introduce a novel visibility metric which computes object visibility in a cumulative manner. We propose an incremental method called *Complete Occlusion-map based Retrieval* (COR) to answer SVK queries. This method exploits effective heuristics to prune the search space and construct a data structure called Occlusion-Map. Then the method adopts the best-first strategy to return relevant objects incrementally. Extensive experiments on real and synthetic data sets suggest that our method is effective and efficient when processing SVK queries.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search Process

General Terms

Algorithms, Experimentation

Keywords

Location-based query, Keyword search, Visibility, Indexing

1. INTRODUCTION

The wide proliferation of powerful smart phones equipped with multiple sensors, 3D graphical engine, and 3G connection has nurtured the creation of a new spectrum of visual mobile applications. In these applications, users in the real world can be supplied "on the spot" by cyber information that is tightly coupled with the physical surroundings. In what people call *cyber-physical mobile applications*, a user holding a smart phone may be able to perform a 3D walkthrough around her location or issue a reality-augmented Web search in her real-time camera video. Such applications identify new data retrieval problems.

Let us look at one example, a tourist in a city of attractions searches for introductory information about a distant grand church within her eyesight. With a reality-augmented search gadget, she can simply type the keyword "church" and then expect the Web pages about *that* particular church to be displayed on her mobile device, as shown in Figure 1. The most distinguishing characteristic of such applications is the need for What-You-Retrieve-Is-What-You-See (WYRIWYS). Specifically, we need to retrieve spatial Web objects which are both visually conspicuous in physical space and semantically relevant in document space.

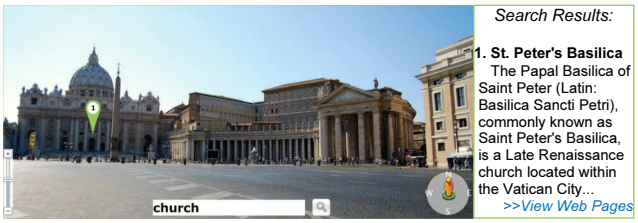


Figure 1: An example of WYRIWYS

Unfortunately, the information retrieval community has not realized the importance of supporting WYRIWYS. State-of-the-art spatial access methods are mostly distance-based and thus inapplicable for supporting WYRIWYS. The recent development of spatial keyword query (SKQ) processing techniques [3, 12, 8, 5, 2] along with several variants [19, 1] can be utilized to support location-based keyword search. However, none of these works can provide WYRIWYS for the following reasons: (1) First, most SKQ techniques overlook the visibility of objects. These techniques model an object as an entity with location and keywords. A typical SKQ processor computes the similarity between an object and the query by combining their spatial proximity and semantic relevancy. Objects that are spatially close to the query location are inclined to be returned, whether they are visible to the user or not. Considering our example of reality-augmented search, a SKQ may retrieve numerous unwanted churches which are close but invisible to the tourist. (2) Second, the previous methods consider each object as a *point of interest*. Such a simplification fails to utilize the extent data of spatial objects, which are readily available among map service

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

providers[1]. Thus, visually conspicuous objects have no privilege over inconspicuous ones when being retrieved.

Motivated by the absence of effective retrieval techniques when developing a cyber-physical mobile search prototype [16], we study a novel query called the *spatio-visual keyword (SVK)* query. Given a collection C of spatial Web objects and a query q, a SVK query returns a list of k objects in C with the largest scores, where the score of each object is computed by combining its physical visibility and semantic relevancy with respect to q. Note that the SVK query is different from simply imposing visible judgment on distance-based spatial keyword queries, since the visual conspicuousness of objects should be quantitatively measured, based on human visual perception. For this purpose, we propose a novel visibility metric, which measures the visible parts of an object in a cumulative way.

The SVK query is interesting not only because it is useful in real applications, but also because it introduces several new challenges. To the best of our knowledge, no state-of-the-art technique exists for the handling of SVK queries. On one hand, instead of being independent, each object may be partially or totally occluded by other objects. Therefore, the processing of SVK queries is much more challenging than that of distance-based spatial keyword queries, considering the complex correlations among the objects. On the other hand, existing works [10, 9, 15] on visible spatial queries employ Boolean judgment to deal with visibility analysis. The major drawback of such a mechanism is that the degree of being visible cannot be distinguished for visible objects. Thus, the existing visible query techniques are not applicable for SVK queries either.

To address the above problems and effectively process SVK queries, we propose the *Complete Occlusion-map based Retrieval* (COR) method. This method employs the hybrid index IR-tree [5] and works in a two-step manner. In the first step, a dynamic structure called *Complete Occlusion-Map* (COM) is built, which partitions the surrounding space of the query point into a number of angular ranges and maintains visibility information for each range. In the second step, COR computes the accurate visibilities for objects, as well as the tightest visibility upper bounds for IR-tree nodes. By taking advantage of the best-first search, the search space is effectively pruned and the top-k relevant objects are returned in an incremental manner.

The main contributions of this paper are:

1. We propose a novel query, namely the SVK query, to retrieve spatial Web objects that are both visually conspicuous and semantically relevant. To the best of our knowledge, it is the first work supporting WYRIWYS for spatial Web objects.
2. We present a quantitative visibility metric, which captures the analytical visual angle and distance of an object with spatial extent.
3. We propose an incremental method, namely COR, to handle SVK queries. And we conduct extensive experiments on both real and synthetic data sets to evaluate the performance of this method.

The rest of this paper is organized as follows. Section 2 reviews existing works on spatial keyword queries and visible spatial queries. Section 3 conducts visibility analysis for spatial Web objects, and formally defines the SVK query. Section 4 provides a baseline solution for SVK query processing. Section 5 presents the detailed COR method. Section 6 empirically evaluates the performance of the proposed solutions. Finally, Section 7 concludes the paper.

[1]http://www.automotiveit.com/bosch-navigation-app-features-3d-"artmap"/news/id-00851

2. RELATED WORK

In this section, we review existing works related to SVK queries, including spatial keyword queries and visible spatial queries.

2.1 Spatial Keyword Queries

Spatial keyword queries have been extensively studied recently. To retrieve objects relevant to the query keywords within a pre-specified region, Chen et al. [3] utilize separate indexes for spatial and textual features. A set of candidate objects is retrieved using the index upon one attribute, which is then refined with the index upon the other attribute. The major drawback of such a mechanism is that the candidate set obtained based on one feature may be too huge. To address this problem, Hariharan et al. [12] propose a hybrid index KR*-tree by augmenting each R*-tree node with a set of keywords appearing in the corresponding subtree. At query time, KR*-tree is traversed, objects containing query keywords and intersecting with the query region are returned. Felipe et al. [8] propose another hybrid index: the IR^2-tree, which integrates R-tree [11] with signature files. In the query process, the signature file of an IR^2-tree node is loaded into memory. Thus, nodes that do not contain all the query keywords are pruned early on and relevant objects are returned incrementally.

Limited by the nature of signature files, the IR^2-tree does not support ranking on textual relevancy. Moreover, loading the signature file of all words into the memory may incur substantial I/O cost. In [5], Cong et al. propose the IR-tree index. Different from IR^2-tree, IR-tree augments each R-tree node with an inverted file for the objects contained in the subtree rooted at that node. Based on IR-tree, Cong et al. develop an efficient solution for answering location-aware top-k text retrieval (LkT) query. Unfortunately, the LkT query is essentially distance-based, which fails to consider object visibility. Thus, objects spatially close to the query location are always inclined to be returned, whether they are visible or not. This difference renders their query techniques infeasible for answering SVK queries.

Several other variants of spatial keyword query have also been studied in [19, 1, 18]. However, these works do not take into account visibilities of objects either, and our focus, the integration of physical visibility and cyber information, has not been studied yet.

2.2 Visible Spatial Queries

Visibility analysis has been well studied in the field of computer graphics [4] and computational geometry [6] in various approaches. To the best of our knowledge, all these approaches assume the data are memory-resident. Apparently, this is not a practical assumption for location-based Web search applications, which usually involve extremely massive data.

In the database community, Kofler et al. [13] propose the LoD-R-tree, which exploits spatial proximity to identify visible objects. Shou et al. [17] design the HDoV-tree, which pre-computes visibility information and incorporates it into the nodes of LoD-R-tree. But the genuine purpose of both LoD-R-tree and HDoV-tree lies in accelerating environment rendering in virtual walkthrough applications. Without maintaining semantic features of objects, they are not capable of supporting SVK queries. More recently, Nutanong et al. propose the visible k nearest neighbor (VkNN) query in [15]. The goal is to retrieve k objects with the minimum smallest visible distances (MINVIDIST) to a query point q. Nutanong et al. prove that it is sufficient to determine the MINVIDIST of an object o by considering only objects nearer than o. On the basis of this observation, they demonstrate the Post-Pruning and Pre-Pruning algorithms to obtain visible nearest neighbors. Several extensions of the VkNN query are also studied [10, 9]. These works do consid-

er the presence of obstacles and the occluding correlations among objects. Nevertheless, the developed techniques do not constitute good solutions for our problem. First, as aforementioned, these works employ Boolean visibility judgment and the degrees of being visible are not distinguished. Second, these works only concern the spatial visibilities of objects, whereas our SVK query focuses on the fusion of both visibility and textual relevancy.

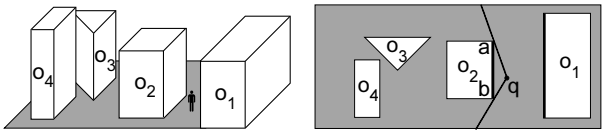
3. PROBLEM DEFINITION

Let $C = \{o_1, o_2, ...o_n\}$ be a collection of spatial Web objects kept by a central server (e.g., a map service provider). Each object $o \in C$ is defined by a tuple $(o.pol, o.height, o.doc)$. Here, $o.pol$ is an arbitrarily shaped polygon, representing the projection of o in the horizontal plane; $o.height$ is the vertical height of the 3D object o; and $o.doc$ is o's Web document. Given a query q with the user location and a set of query keywords specified, i.e., $q = (q.location, q.keywords)$, a SVK query retrieves objects in C that are both visually conspicuous and semantically relevant to q. In the following, we first present a quantitative visibility metric in Section 3.1, then combine it with textual relevancy to propose a hybrid ranking mechanism in Section 3.2, based on which we formulate the SVK query in Section 3.3.

3.1 Visibility Analysis

A number of approaches have been proposed in the computer graphic community to efficiently render visible objects in 3D environments. These approaches assume the 3D models of objects are memory-resident, and exploit graphical techniques (e.g., hierarchical Z-buffer) to clip out hidden polygons on the surface of an object so that visible scenes can be rendered vividly on graphical units. However, these approaches are inapplicable for visibility computation in SVK applications: Given the huge number of spatial Web objects, it is extremely difficult, if not impossible, for the server to access all the complex 3D details of objects, and meanwhile ensure instantaneous response to the query launcher. Hence, there is an evident necessity of simplifying occlusion detection among 3D objects, in such a way as to alleviate the computation burden on the server, but simultaneously guaranteeing a considerable precision.

In view of this problem, we employ a strategy which has been widely adopted in walkthrough systems [4]. This strategy consists of two steps to determine the visibility of each object. First, it utilizes the horizontal (2D) projections of objects to determine occlusion. Second, for each unobstructed object, it uses an accurate 3D algorithm to compute visibility. The rationale behind the 2D approximation of the first step is that, in a wide range of SVK scenarios, the heights of objects are actually in the same range of magnitude, and thus play a less important role in determining if an object is visible or not. Just as shown in Figure 2(a), in a typical urban area, it is common that the densely distributed buildings are not significantly distinguishable in height with respect to an observer. Moreover, since nearer buildings always appear taller than faraway ones, it is safe to judge whether an object is visible by examining its horizontal projection (see Figure 2(b)).



(a) Side view (b) Top view

Figure 2: An urban area with four buildings

Relying on the above assumption, we introduce the concepts of unobstructed segment and unobstructed face to facilitate visibility computation for objects.

DEFINITION 1. *C is a collection of spatial Web objects and q is a query point. For any object $o \in C$, ab is an unobstructed segment of o iff the following two conditions hold: 1) ab is on the boundary of $o.pol$, 2) for any point p on ab, segment pq does not cut the projected polygon of any object.*

Informally, an unobstructed segment is a segment on the boundary of the projected polygon that can be completely seen by the observer. For example, in Figure 2(b), the unobstructed segments of objects $o_1...o_4$ are represented by bold lines. Take object o_2 as a specific example, segment ab is unobstructed because any point on ab is not occluded by obstacles. It is not difficult to find that, each unobstructed segment in the horizontal plane actually determines an unobstructed face of the object in the 3D space. Below, we give the formal definition of unobstructed face.

DEFINITION 2. *Given an object $o \in C$, each unobstructed segment ab of o determines an unobstructed face F_{ab}, which is a vertical rectangle of length $|ab|$ and height $o.height$.*

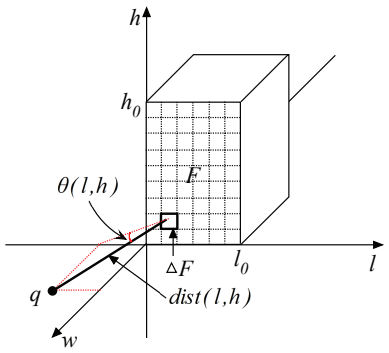


Figure 3: Cumulative face visibility

We proceed to analyze the visibility of an unobstructed face. Intuitively, given an unobstructed face F, F appears more conspicuous if it is *large* and *close* to the observer. However, the challenge is how to define being *large* and *close*, given the fact that different parts on an unobstructed face always have different distances and orientations. To handle this, we develop a cumulative computing approach. As shown in Figure 3, we divide F into numerous infinitesimal grids. For each grid, say ΔF, the distances as well as orientations of all points on ΔF to q are considered to be equal due to its infinitesimal size. Based on this property, we represent the visibility of ΔF as

$$vis(q, \Delta F) = \frac{area_{\Delta F} \cdot \sin\theta_{\Delta F}}{dist^2_{\Delta F}} \quad (1)$$

Here, $area_{\Delta F}$ reflects how *large* ΔF is; $\theta_{\Delta F}$ is the angle between $\overrightarrow{q\Delta F}$ and F, measuring the orientation of ΔF with respect to the observer (when $\theta = 90^\circ$, ΔF right faces q); $dist_{\Delta F}$ is the Euclidean distance between q and ΔF which tells how *close* ΔF is. Since ΔF is a 2D grid, its visibility is inversely proportional to $dist^2_{\Delta F}$ instead of $dist_{\Delta F}$.

Based on Equation 1, the visibility of the entire unobstructed face F can be obtained by summing up the visibilities of all the small grids, as defined in Definition 3.

DEFINITION 3. *Given a query point q and an unobstructed face F, for any point $p = (l, h) \in F$, let $dist(l, h)$ be the Euclidean distance between q and p, and $\theta(l, h)$ be the angle between $\overrightarrow{qp}$ and F, then the visibility of F with respect to q is defined as*

$$vis(q, F) = \iint_F \frac{\sin\theta(l,h)}{dist^2(l,h)} \mathrm{d}l\mathrm{d}h. \quad (2)$$

The object visibility can then be defined by summing up visibilities of all the unobstructed faces belonging to it.

DEFINITION 4. *Given an object o, let $S(o)$ be the set containing all the non-overlapping unobstructed faces of o, then the visibility of o with respect to query q is defined as*

$$vis(q, o) = \sum_{F \in S(o)} vis(q, F).$$

3.2 Ranking Mechanism

To measure the semantic relevancy between a spatial Web object and a user query, we adopt the classic TF-IDF model, which has been widely used in traditional search engines. According to the TF-IDF model, an object o is assigned a higher ranking value if the query keyword occurs frequently in $o.doc$ and infrequently in documents of other objects. Below, we give the formal definition of semantic relevancy, based on which we derive the weighted ranking score for a spatial Web object.

DEFINITION 5. *Given a collection of spatial Web object C and a query q, for any object $o \in C$, the semantic relevancy between q and o is:*

$$rel(q, o) = \sum_{t \in q.keywords} TF(t, o.doc) \cdot IDF(t, C)$$

where

$$TF(t, o.doc) = 1 + \ln(frequency(t, o.doc))$$

$$IDF(t, C) = \ln \frac{|C|}{1 + |objects\ in\ C\ that\ contain\ t|}.$$

DEFINITION 6. *Given a collection of spatial Web object C, a query q, and a pre-specified value $\alpha \in [0, 1]$, for any object $o \in C$, the weighted ranking score of o with respect to q is:*

$$score(q, o) = \alpha \cdot ||vis(q, o)|| + (1 - \alpha) \cdot ||rel(q, o)||.$$

The weighted ranking score is virtually a linear interpolation that combines physical visibility and semantic relevancy, and α is a parameter used to balance two aspects. Note that both visibility and relevancy of an object are normalized into the range $[0, 1]$.

3.3 The SVK Query

Based on the definition of weighted ranking score, we formulate the SVK query as follows.

DEFINITION 7. *Given a collection of spatial Web objects C, a user query q, and an integer k, the result of SVK query is a list L of spatial Web objects such that, 1)$L \subseteq C$, 2)$|L| = k$ (given that $|C| \geq k$), 3)$\forall o \in L, \forall o' \in C - L, score(q, o) \geq score(q, o')$.*

To specify our previous example query for church, Figure 4 depicts the top view of an area in Rome comprising ten buildings $o_1 ... o_{10}$, whose documents are described in Table 1. A tourist in this area may want some introductory information when he faces a grand church o_1, and thus issues a top-1 SVK query at q with the keyword "church". As a result, church o_1 with its cyber information will be returned to the user, since it is both visually conspicuous and semantically relevant. Note that, although another church o_6 is actually nearer than o_1, it is not returned because it is completely occluded by o_7 and is not what the user demands.

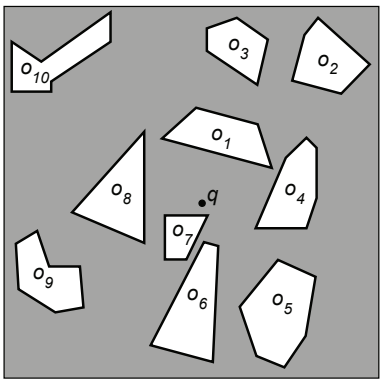


Figure 4: Top view of ten spatial Web objects

Table 1: Documents of objects $o_1 ... o_{10}$

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
Gothic	5	1	4	2	0	5	0	0	0	0
Church	5	2	0	0	0	5	0	0	0	0
Museum	0	0	0	0	3	0	4	2	4	5

4. A BASELINE SOLUTION

In this section, we provide a baseline solution for answering SVK queries. The idea is quite straightforward. Given a spatial Web object, its ranking score is a combination of its visibility and semantic relevancy. Hence, we can exploit separate spatial and textual indexes of objects, and adapt the threshold algorithm [7] to return top-k results. Algorithm 1 gives an overview of the solution.

Algorithm 1: $Baseline(q, \alpha, k, RTree, InvertedFile)$

```
1  Build a list L of all spatial web objects;
2  Sort L in the descending order of semantic relevancy;
3  Initialize an empty set A of a fixed size k;
4  foreach object o in L do
5      threshold = the least ranking score in A;
6      if |A| = k and score_max(q, o) ≤ threshold then
7          break;
8      Compute vis(q, o) with RTree;
9      score(q, o) = α · ||vis(q, o)|| + (1 − α) · ||rel(q, o)||;
10     if |A| < k or score(q, o) > threshold then
11         Update A with o;
12 return A
```

As shown, we first build a list L of all spatial Web objects, which are sorted in the descending order of semantic relevancy. Then we initialize a empty result set A with a fixed size k, and keep track of the least ranking score in A as $threshold$. Next, we scan L sequentially and progressively update A (lines 4-11). Note that L is actually unnecessary to be scanned entirely. For object o currently being processed, if o's score is below $threshold$ even when we set o's visibility to the visibility upper bound (line 6), it is ensured further scanning will not generate top-k results any more and the algorithm safely terminates.

Clear as Algorithm 1 appears, the difficult part lies in the computation of object visibility. In the following, we elaborate line 8 and discuss how to derive object visibility with R-tree. Note that the R-tree only needs to index the projected 2D polygons of objects, with their heights stored as associating features to support visibility computation.

In the horizontal plane, each object o occupies an angular range with respect to the query point q. Apparently, only objects falling in that angular range and nearer than o can potentially occlude o. We retrieve such objects using R-tree and call them *potential occluders*. Next, we need to retrieve o's unobstructed segments, and then sum up the visibilities of their corresponding unobstructed faces. Recall Definition 1, a segment ab is identified as unobstructed when any point on ab is not occluded by other objects in the horizontal plane. In other words, an unobstructed segment is essentially the nearest segment to the observer within its angular range. Thus, the problem is equivalent to comparing edges of $o.pol$ with polygon edges of the *potential occluders*, and obtain the nearest segments that belong to $o.pol$. Suppose e_1 and e_2 are two edges sharing a common angular range $[\theta_{low}, \theta_{high}]$, and their portions in $[\theta_{low}, \theta_{high}]$ are ab and cd respectively. Then, as shown in Figure 5, the relationship of segments ab and cd can be categorized into four cases:

- Case 1: ab and cd completely overlap in $[\theta_{low}, \theta_{high}]$.
- Case 2: ab and cd do not intersect in $[\theta_{low}, \theta_{high}]$.
- Case 3: ab and cd intersect, and the intersecting point is along θ_{low} or θ_{high}.
- Case 4: ab and cd intersect, and the intersecting point is along $\theta \in (\theta_{low}, \theta_{high})$.

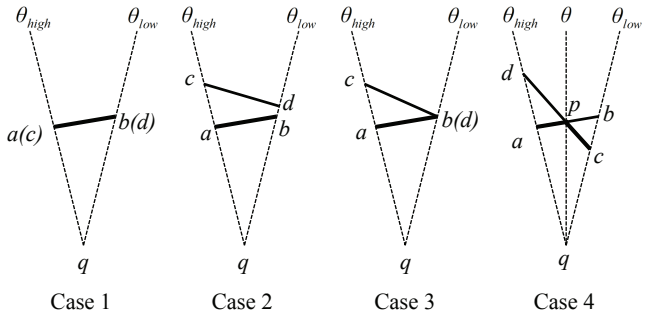


Figure 5: Edge relationships

Case 1, 2 and 3 are actually the same to deal with. In the horizontal plane, as long as one segment is nearer than the other along any direction $\theta \in (\theta_{low}, \theta_{high})$, it is guaranteed that segment will be nearer for the entire angular range. For Case 4, both segments are nearer to the query point but in different sub angular ranges. We need to compute the intersecting point p and split $[\theta_{low}, \theta_{high}]$ along the direction of $\overrightarrow{qp}$. Then the nearer segment in each sub range can be easily obtained.

By comparing the polygon edges of o and o's *potential occluders*, we are able to obtain the nearest segments in o's angular range, and identify o's unobstructed segments. Then the unobstructed faces of o in the 3D space can be constructed. The remaining job is to compute visibility for each unobstructed face and sum them up. For clarity but without loss of generality, we assume the coordinate of the query point q in the 3D space is $(x_q, 0, 0)$ [2]. Given an unobstructed face F constituted by four points $(0, 0, 0)$, $(0, l_0, 0)$, $(0, l_0, h_0)$ and $(0, 0, h_0)$, the visibility of F with respect to q can be derived as follows.

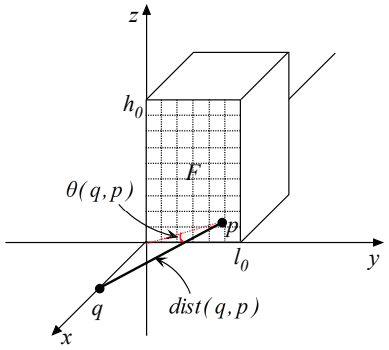


Figure 6: Unobstructed face visibility computation

Refer to Figure 6, for any point $p \in F$, we denote by $\theta(q, p)$ the angle between $\overrightarrow{qp}$ and F, by $dist(q, p)$ the distance between q and p, and by $dist(q, F)$ the perpendicular distance from q to face F. Then the following equation holds.

$$\sin\theta(q,p) = \frac{dist(q,F)}{dist(q,p)} \tag{3}$$

Combining Definition 3 and Equation 3, the visibility of F can be further developed as follows:

$$\begin{aligned} vis(q,F) &= \iint_{\substack{0\le y\le l_0\\ 0\le z\le h_0}} \frac{dist(q,F)}{dist^3(y,z)}\mathrm{d}y\mathrm{d}z \\ &= \int_0^{h_0}\int_0^{l_0} \frac{x_q}{(x_q^2+y^2+z^2)^{\frac{3}{2}}}\mathrm{d}y\mathrm{d}z \\ &= \int_0^{h_0} \frac{x_q l_0}{(z^2+x_q^2)\sqrt{z^2+x_q^2+l_0^2}}\mathrm{d}z \\ &= \arctan\frac{l_0 h_0}{x_q\sqrt{x_q^2+l_0^2+h_0^2}} \end{aligned}$$

We can see the visibility of an unobstructed face is closely related to: (1) its area, (2)its minimum visible distance to the query point, and (3) its maximum visible distance to the query point. This explains why the cumulative visibility provides an accurate capture of the visual features of an object.

The above discussion addresses the problem of computing object visibility and thus makes Algorithm 1 feasible. Unfortunately, the baseline solution is far from being efficient. On one hand, since an unobstructed segment is the only occluder in its angular range, it is highly redundant to retrieve *potential occluders* repeatedly to compute visibility for each object in L. On the other hand, the separation of indexes fails to closely combine the spatial and textual features of objects, and incurs unnecessary overhead.

5. THE COR METHOD

In this section, we present the COR method to handle SVK queries more effectively. This method employs the hybrid index IR-tree, and performs query processing in two steps. In the first step, a novel structure called Complete Occlusion-Map (COM) is constructed, underpinning the support for high-performance visibility computation. In the second step, the method uses the best-first strategy to retrieve the top-k relevant objects incrementally. In what follows,

[2]Handling the cases where $y_q, z_q \neq 0$ is straightforward using the substitution method for integral.

we first give a brief introduction to the IR-tree index and the COM structure in Section 5.1, then elaborate the two steps in Section 5.2 and Section 5.3, and finally discuss the efficiency of the COR method in Section 5.4.

5.1 Data Structures

5.1.1 The IR-Tree Index

IR-tree [5] is essentially an extension of R-tree by augmenting each node with a pointer to an inverted file. For a leaf node N_L, its inverted file indexes the documents of all objects in N_L. For an intermediate node N_I, its inverted file indexes the pseudo documents of its child nodes. The pseudo document is an important concept to facilitate the computation of textual relevancy upper bound, because it summarizes the terms of all child entries. Figure 7 is an example IR-tree for the spatial Web objects $o_1 ... o_{10}$ shown in Figure 4. The ten polygon-shaped objects are grouped according to their spatial proximity to build up the IR-tree. Similar to the case of R-tree, as we detect object occlusion in the horizontal plane, the IR-tree needs to index only the projected polygons of objects, and stores their corresponding heights as additional feature information. Table 2 lists inverted file contents for all IR-tree nodes.

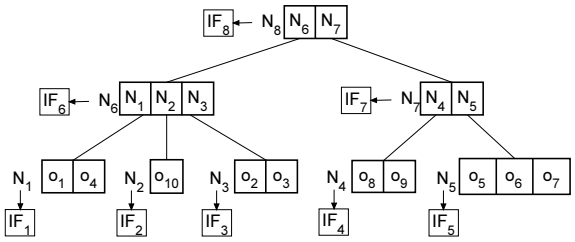


Figure 7: The IR-tree

Table 2: IR-Tree Inverted Files

	Gothic	Church	Baroque
IF_1	$(o_1, 5), (o_4, 2)$	$(o_1, 5)$	
IF_2			$(o_{10}, 5)$
IF_3	$(o_2, 1), (o_3, 4)$	$(o_2, 2)$	
IF_4			$(o_8, 2), (o_9, 4)$
IF_5	$(o_6, 5)$	$(o_6, 5)$	$(o_5, 3), (o_7, 4)$
IF_6	$(N_1, 5), (N_3, 4)$	$(N_1, 5), (N_3, 2)$	$(N_2, 5)$
IF_7	$(N_5, 5)$	$(N_5, 5)$	$(N_4, 4), (N_5, 4)$
IF_8	$(N_6, 5), (N_7, 5)$	$(N_6, 5), (N_7, 5)$	$(N_6, 5), (N_7, 4)$

5.1.2 Complete Occlusion-Map

In the horizontal plane, each unobstructed segment occupies an angular range and causes an invisible region with respect to the observer. Moreover, the angular ranges of different unobstructed segments never overlap. Therefore, given a query point q and a collection C of spatial Web objects, the set of all unobstructed segments from C in fact partitions the space surrounding q into a number of angular ranges, each corresponding to one or zero unobstructed segment.

As shown in Figure 8, $[\theta_{qb}, \theta_{qa}]$ is an angular range occupied by unobstructed segment ab. In $[\theta_{qb}, \theta_{qa}]$, it is guaranteed that only the corresponding unobstructed face F_{ab} is visible and all other objects are hidden behind ab, i.e., the angular range $[\theta_{qb}, \theta_{qa}]$ is dominated by F_{ab}. We call $[\theta_{qb}, \theta_{qa}]$ a dominated angular range, and define a tuple (SEG, OID, VIS, ANC) to describe it, where, SEG denotes the dominating unobstructed segment, VIS is the visibility of the unobstructed face F_{SEG}, OID is the id of the object that SEG belongs to, and ANC is an array storing the ancestor node ids of object o_{OID} in the IR-tree. Later in Section 5.3 we will see,

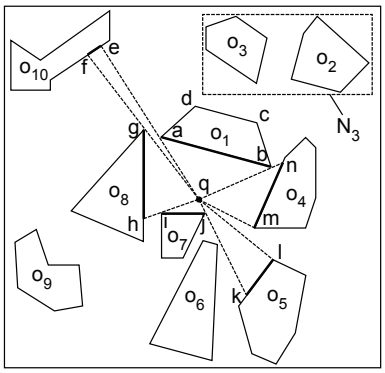


Figure 8: Partitioning the surrounding space of q into angular ranges

these attributes facilitate the visibility computation in the dominated angular range.

In Figure 8, $[\theta_{ql}, \theta_{qm}]$ is an angular range empty of unobstructed segment. Such an angular range indicates no object from C lies in it. We call it a dummy angular range.

In summary, given a collection of spatial Web objects C and a query point q, let $S(C)$ be the set of all unobstructed segments from C, then the COM of q is a partition of the space by $S(C)$ into a set of angular ranges, each angular range is either dominated or dummy. Table 3 illustrates the COM for the ten objects shown in Figure 8.

Table 3: Complete Occlusion-Map

AR	SEG	OID	VIS	ANC
$[\theta_{qb}, \theta_{qa}]$	ba	1	$vis(q, F_{ba})$	(8, 6, 1)
$[\theta_{qe}, \theta_{qf}]$	ef	10	$vis(q, F_{ef})$	(8, 6, 2)
$[\theta_{qg}, \theta_{qh}]$	gh	8	$vis(q, F_{gh})$	(8, 7, 4)
$[\theta_{qi}, \theta_{qj}]$	ij	7	$vis(q, F_{ij})$	(8, 7, 5)
$[\theta_{qk}, \theta_{ql}]$	kl	5	$vis(q, F_{kl})$	(8, 7, 5)
$[\theta_{ql}, \theta_{qm}]$	null	null	0	$\emptyset$
$[\theta_{qm}, \theta_{qn}]$	mn	4	$vis(q, F_{mn})$	(8, 6, 1)

5.2 Construction of Complete Occlusion-Map

In this subsection, we discuss the construction of COM. The idea is to traverse the IR-tree and retrieve all the unobstructed segments to partition the space. A simple search process is described as following: 1) Initialize COM with a large dummy angular range covering the entire plane. 2) Create a priority queue PQ with the IR-tree root node. For each entry in PQ, its MINDIST to the query point q is used as its key. 3) If the head entry of PQ is a node, dequeue it and insert all its child entries into PQ. 4) If the head of PQ is an object, dequeue it and update current COM. To be more specific, each projected polygon edge of the object is compared with the current COM components to produce nearer segments. 5) Repeat step 3) and 4) iteratively. When PQ is empty, the construction finishes.

Unfortunately, the above process needs to access all IR-tree nodes and examine every polygon edge of an object. In the following, we elaborate the construction process by presenting three heuristics to effectively prune the search space.

Firstly, when incorporating an object o to update COM, one can verify that some edges are in fact hidden by other edges of $o.pol$ itself. As a concrete example, for o_2 in Figure 8, its edges bc, cd, and da are occluded by edge ab. Apparently, these edges cannot be unobstructed and it is safe to skip them. Based on this observation,

we introduce the concept of back edge, and derive Heuristic 1 to avoid unnecessary edge comparisons.

DEFINITION 8. *Given a spatial Web object o and a query point q, an edge e of $o.pol$ is a back edge iff for any point p on e, segment pq cuts at least one other edge of $o.pol$.*

HEURISTIC 1. *C is a collection of spatial Web objects and q is a query point. For any object $o \in C$, none of the back edges of o can be unobstructed to q.*

The second optimization aims to avoid unnecessary node expansions. Refer to Figure 8 again, N_3 is an IR-tree node bounding $o_2.pol$ and $o_3.pol$. During the construction of COM, due to a smaller MINDIST, $o_1.pol$ is examined before N_3 and segment ab is retrieved. As a result, N_3's MBR is completely occluded by segment ab. Thus, neither o_2 nor o_3 can be visible and N_3 can be pruned.

HEURISTIC 2. *During the construction of COM, if N is an IR-tree node satisfying the following two conditions: 1) the query point q stays outside N's MBR, and 2) N's MBR is completely occluded by current COM. Then no object in N may have unobstructed segment.*

In the horizontal plane, each unobstructed segment in COM could be interpreted as an enhanced nearest surrounder [14], which is associated with visibility and ancestor IR-tree nodes. Therefore, a condition designed in [14] to early terminate nearest surrouder query also applies in COM construction.

HEURISTIC 3. *The construction of COM can terminate when the following two conditions hold: 1) current COM does not have dummy angular ranges, 2) the MINDIST of the head entry in PQ is larger than the minimum bounding radius of current COM.*

Algorithm 2: $Construct\ COM(q, IRTree)$

```
1  Initialize COM to be dummy;
2  Initialize a priority queue PQ with IRTree.root;
3  while !PQ.isEmpty() do
4      r = minimum bounding radius of COM;
5      if !COM.hasDummy() and PQ.head().key ≥ r then
6          break;
7      E=PQ.pophead();
8      if E is an object then
9          Update COM with non-back edges of E.obj;
10     else
11         foreach entry e in E do
12             if e.MBR is not completely occluded then
13                 key = MINDIST(q, e.MBR);
14                 newEntry=(e,key);
15                 Insert newEntry into PQ;
16 return COM
```

With the above three heuristics, we give the elaborated process for COM construction in Algorithm 2. As shown, entries of the IR-tree are accessed in the ascending order of their minimum distances to q. In this way, nearby objects, which may bring about larger invisible space, are given priority. Hence, the prunable area keeps shrinking and a large number of upcoming entries are pruned (Heuristic 2). Moreover, so long as the currently retrieved segments completely surround q and that their bounding radius is smaller than the MINDIST of the head entry in PQ. The construction terminates successfully according to Heuristic 3.

5.3 Incremental Object Retrieval

In this subsection, we discuss the computation of accurate ranking scores for objects and ranking score upper bounds for IR-tree nodes, then propose an efficient algorithm to retrieve top-k objects incrementally.

5.3.1 Object Ranking Score

Given an object o, its ranking score is the linear combination of its visibility and semantic relevancy. The semantic relevancy is quite straightforward to derive by Definition 5, as the terms of $o.doc$ are all maintained in the IR-tree. Below, we mainly discuss the computation of its accurate visibility.

The basic idea is to use the COM structure to select all unobstructed faces that belong to o, and then add up their visibilities. Relying on the OID attribute, the selection is easy, as exemplified in the following using Figure 8. In the horizontal plane, object o_{10} intersects with three COM angular ranges: $[\theta_{qb}, \theta_{qa}]$, $[\theta_{qe}, \theta_{qf}]$, and $[\theta_{qg}, \theta_{qh}]$. In $[\theta_{qb}, \theta_{qa}]$, the value of OID is 1, indicating that the unobstructed face must belong to object o_1 and o_{10} is occluded. The case is similar in $[\theta_{qg}, \theta_{qh}]$ where $OID = 8$. However, in $[\theta_{qe}, \theta_{qf}]$, OID is 10. We conclude that the corresponding unobstructed face F_{ef} must belong to o_{10}, and thus return $vis(q, F_{ef})$ as o_{10}'s visibility.

5.3.2 Node Ranking Score Upper Bound

We proceed to discuss visibility upper bound computation for IR-tree nodes. The key is to utilize the attribute ANC of the COM structure. Given an IR-tree node N and a dominated COM angular range intersecting N's MBR, the corresponding unobstructed face belongs to N if and only if N's id is contained in ANC. Hence, the computation of visibility upper bound for node N is performed as following: 1) Obtain all COM components that intersect N's MBR. 2) Find the dominated components whose ANCs contain N's id, and suppose they are called $candidates$. Please note that all dummy components are ignored here, because no object in N can actually lie in the dummy angular ranges and the occupation must be caused by MBR expansion. 3) Group the $candidates$ by their OIDs, thus unobstructed faces of the same object are gathered into the same group. 4) Aggregate the VIS in each group to obtain the group visibility. 5) Find the maximum visibility among all groups and return it as the visibility upper bound of N, which is denoted by $vis(q, N)$.

THEOREM 1. *The visibility obtained from the above process is a tightest visibility upper bound for node N.*

PROOF. Step 1), 2), and 3) actually find all the visible objects in N. By computing their visibilities in step 4) and comparing them in step 5), it is guaranteed that no object in N has a larger visibility than $vis(q, N)$, but at least one object reaches it. □

The semantic relevancy between q and N can be derived from N's pseudo document. Recall that for any term t, the weight of t in N's pseudo document is no smaller than that of any object in N.

Combining the visibility upper bound $vis(q, N)$ and relevancy upper bound $rel(q, N)$, N's weighted ranking score upper bound is given by:

$$score(q, N) = \alpha \cdot ||vis(q, N)|| + (1 - \alpha) \cdot ||rel(q, N)|| \quad (4)$$

THEOREM 2. *The score computed from Equation 4 is a ranking score upper bound for N.*

PROOF. For any object o contained in N, the following inequality holds according to Theorem 1:

$$vis(q, N) \geq vis(q, o)$$

And according to the property of pseudo document, we have:

$$rel(q, N) \geq rel(q, o)$$

Thus:

$$score(q, N) \geq \alpha \cdot ||vis(q, o)|| + (1-\alpha) \cdot ||rel(q, o)|| = score(q, o)$$

which completes the proof. □

5.3.3 Retrieval Algorithm

Theorem 2 ensures a hierarchical order of IR-tree nodes' ranking scores. Now, we present the algorithm for object retrieval, which adopts the best-first strategy and returns top-k objects in an incremental manner. As shown in Algorithm 3, a priority queue PQ is maintained to store objects and IR-tree nodes to be visited, and is initialized with the root node of IR-tree. The ranking score of each entry is used as its key in PQ. If the head entry of PQ is possessed by an IR-tree node, we expand it and elaborate the ranking scores of its children. If the head entry is occupied by a spatial Web object o, it indicates that all the objects in PQ do not have a larger score than o, thus it is safe to report o as the next most relevant object. When the top-k objects are obtained or PQ becomes empty, the algorithm terminates successfully.

Algorithm 3: $Object\ Retrieval(q, \alpha, k, COM, IRTree)$

```
1  Initialize an empty result set A;
2  Initialize a priority queue PQ with IRTree.root;
3  while !PQ.isEmpty() and |A| < k do
4      E=PQ.pophead();
5      if E is an object then
6          Insert E.obj into A;
7      else
8          foreach entry e in E do
9              score = α · ||vis(q, e)|| + (1 − α) · ||rel(q, e)||;
10             newEntry=(e,score);
11             Insert newEntry into PQ;
12 return A
```

5.4 Discussion

In the COR method, the cost of the first step is closely related to the environment around the query point q. If q lies in a dense area, the COM construction will be quite efficient. This is because unobstructed segments can be retrieved early to surround q, thus most IR-tree nodes can be pruned with Heuristic 2 and 3. On the contrary, if q lies in an extremely sparse area, a considerable number of IR-tree nodes may need to be accessed. The cost of the second step is expected to be small. On one hand, as proved in Theorem 1, the visibility upper bound for IR-tree node is optimal. On the other hand, the tightness of the upper bound for semantic relevancy depends on the number of query keywords. Since fewer keywords indicate tighter upper bound for semantic relevancy and in most applications a query contains only limited number of keywords, the relevancy upper bound is also expected to be tight.

6. EXPERIMENTS

In this section, we empirically evaluate the effectiveness and efficiency of the Baseline and COR methods. We implement both algorithms in JAVA and conduct the experiments on an Intel Core 2 Duo 2.93Ghz PC with 2GB memory.

6.1 Data Sets

Our experiments are based on both real and synthetic data sets. Similar to [5], the real spatial data set LA[3] is used. LA contains 131,461 MBRs, representing street objects in Los Angeles. We normalize all MBRs in LA into the [0,10000]×[0,10000] horizontal space, and assign a height within [10,20] to each MBR. Meanwhile, we have crawled a real textual data set from Gowalla[4], which is a location-based social network providing checking-in and commenting services. The crawled Gowalla set consists of 28,867 Web documents. Each document is a collection of user-generated comments that aim to describe a point of interest in Los Angeles. By randomly selecting a document from the Gowalla set for each spatial object in the LA set, we obtain a spatial-textual data set, named REALDATA.

We also exploit four synthetic data sets to evaluate scalability of the methods, with a cardinality of 100k, 200k, 500k, and 1,000k. Each object in a synthetic data set is described by a tuple $(o.mbr, o.height, o.doc)$. Here, $o.mbr$ is generated uniformly in the horizontal space with a size no larger than 4.00×4.00 (length×width), $o.height$ is within [10,20], and $o.doc$ is selected randomly from the Gowalla set.

We use disk-based R-tree (for Baseline) and IR-tree (for COR) to index these data sets. The page size is fixed at 4KB and the maximum number of branches per node is 100.

6.2 Query Result Demonstration

We first issue some test queries, and see if the proposed methods can indeed retrieve spatial Web objects that are both conspicuous and relevant. For comparison, we also implemented the LkT algorithm [5], which is designed for distance-based spatial keyword retrieval. Figure 9 and Table 4 report the results of three test queries on REALDATA. Specifically, Figure 9(a) shows the 131461 objects in REALDATA as well as the locations of the three queries. Figure 9(b), 9(c) and 9(d) are the respective zoomed-in results of each query, where the green rectangles are the top-5 SVK query results with the inside letters denoting object ids. And we use blue rectangles to highlight objects that are in the top-5 LkT query results but not in the top-5 SVK list.

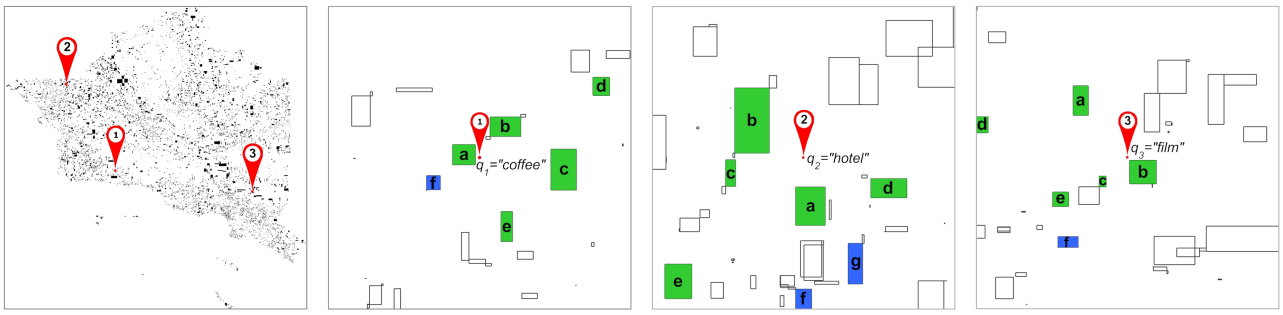
As shown, the SVK query can retrieve objects that are physically visible and semantically relevant. Taking the first query as a concrete example, the returned five objects are all within the user's eyesight (see Figure 9(b)) and closely related to "coffee" (see Table 4). Such query results can serve the user with useful cyber information regarding the physical world that the user sees, and thus provide an immersive searching experience. Please note that although object f is semantically relevant, it does not appear in the top-5 SVK query results due to low visibility. In contrast, the LkT query always tends to retrieve objects that are close to the user, even if they are totally invisible. This is because LkT does not take into account object visibility and thus cannot provide WYRIWYS.

6.3 Efficiency Study

In this subsection, we report the efficiency of Baseline and COR under various parameter settings. Table 5 shows the parameter ranges, where the numbers in bold denote the default settings. For each experiment set, we evaluate the effect of one parameter while the others are fixed at their default values, and run 100 randomly generated queries with their average cost reported. Two performance metrics are employed, i.e., the query time and the number

[3]http://www.rtreeportal.org

[4]http://en.wikipedia.org/wiki/Gowalla



(a) 3 queries on REALDATA (b) Query result 1 (c) Query result 2 (d) Query result 3

Figure 9: Three test SVK queries on REALDATA and their results

Table 4: A Comparison of SVK and LkT Query Results ($\alpha = 0.5$ $k = 5$)

Query	Top-5 SVK Query Results			Top-5 LkT Query Results		
	Object ID	Object Name	SVK Score	Object ID	Object Name	LkT Score
q_1="coffee"	a	Frances Bakery & Coffee	0.806	f	Starbucks	0.985
	b	McDonald's	0.729	a	Frances Bakery & Coffee	0.966
	c	Strawberry Restaurant	0.636	b	McDonald's	0.818
	d	Costa Coffee	0.529	d	Costa Coffee	0.798
	e	Cole's Restaurant	0.519	c	Strawberry Restaurant	0.743
q_2="hotel"	a	Radisson Hotel	0.946	a	Radisson Hotel	0.975
	b	Rose Bowl Motel	0.874	b	Rose Bowl Motel	0.948
	c	Holiday Hotel	0.817	c	Holiday Hotel	0.853
	d	Pasadena Inn	0.649	g	Traveler Motel	0.840
	e	Vegabond Hotel	0.521	f	Hudson Hotel	0.759
q_3="film"	a	Pacific Theater	0.791	a	Pacific Theater	0.846
	b	Park Hotel	0.619	e	Nunavo Cinema	0.797
	c	Highland Club	0.518	f	Alex Theater	0.764
	d	Keda Entertainment	0.512	c	Highland Club	0.619
	e	Nunavo Cinema	0.508	b	Park Hotel	0.613

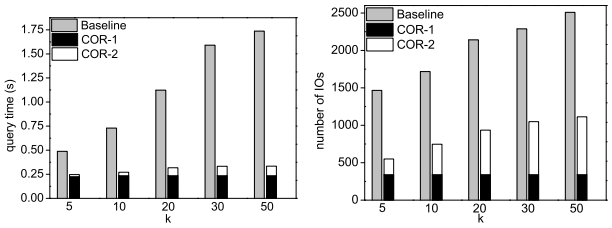
of IOs. In the following, we present the efficiency results and our findings. Unless stated explicitly, REALDATA is used.

Table 5: Parameter ranges and default values

Parameter	Range
k	5, **10**, 20, 30, 50
α	0.1, 0.3, **0.5**, 0.7, 0.9
number of keywords	**1**, 2, 3, 4

6.3.1 *Effect of k*

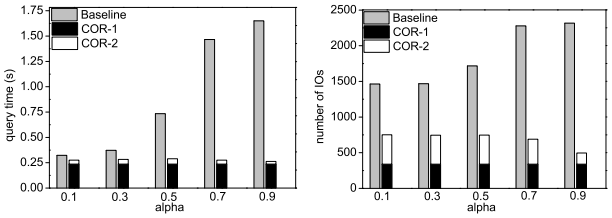
In the first set of experiments, we study the effect of k on the performance of the two methods, and Figure 10 shows the results. Since COR works in a two-step manner, its cost is broken into two parts: COR-1 denotes the cost of occlusion-map construction and COR-2 denotes the cost of top-k objects retrieval. As expected, COR significantly outperforms Baseline in terms of both query time and I/O cost. The total cost of both methods increase with k, since a larger k causes a larger search space to obtain relevant objects. But for COR, the cost of occlusion-map construction is irrelevant to k. Another interesting finding for COR is that, although the I/O cost of object retrieval step is larger than that of occlusion-map construction step, the computation cost is smaller. It is explained by the cheap CPU cost of the most frequently used id-comparison operations in the retrieval step.



(a) query time (b) IO cost

Figure 10: The effect of k

However, Baseline sequentially scans all objects based on semantic relevancy, and thus suffers from a large number of visibility computations.



(a) query time (b) IO cost

Figure 11: The effect of α

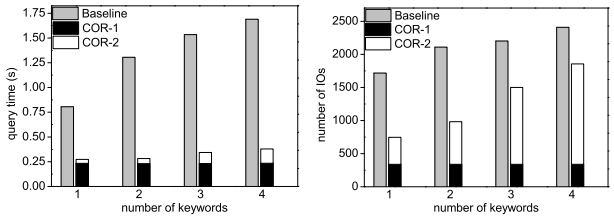
6.3.2 *Effect of α*

In the second experiment set, we study the effect of α and report the results in Figure 11. Again, we can see COR performs much better than Baseline. Moreover, Baseline deteriorates dramatically as α increases, whereas COR performs well stably. The reason behind is that, a large α means more emphasis is put on visibility.

6.3.3 *Effect of number of keywords*

In the third experiment set, we look into the effect of the number of query keywords. As shown in Figure 12, both methods perform

worse with an increasing number of query keywords. For Baseline, a large number of query keywords implies that many objects are not distinguished in terms of semantic relevancy, thus the threshold algorithm terminates later and more objects need to be examined. For COR, although the cost of occlusion-map construction is not related to the number of query keywords, the retrieval step incurs additional overhead. Specifically, as the number of keywords increases, the relevancy upper bound becomes looser. Therefore, more IR-tree nodes need to be accessed, which elevates the number of IOs. Fortunately, as the ranking score computation is quite cheap, the query time increases slowly.

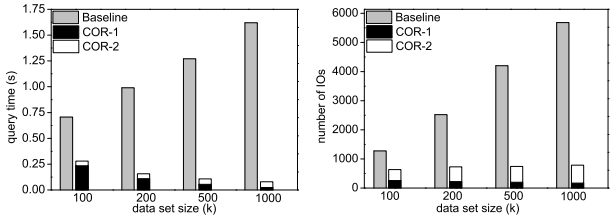


(a) query time (b) IO cost

Figure 12: The effect of number of keywords

6.3.4 Results for synthetic data sets

We have also carried out extensive experiments on the synthetic data sets to evaluate the effects of different parameters. We observe that the tendencies are quite similar to that of REALDATA. Thus, we only report a subset of them to demonstrate the scalability of the two methods. Figure 13 shows the query time and IO cost for experiments on different data sets, when all parameters are fixed at their default values. We observe that, Baseline performs worse with an increasing data set size, whereas COR benefits from it. This is expected, the performance improvement is from the occlusion-map construction step. When the number of objects is large, the query point is more likely to lie in a dense area, thus unobstructed segments can be retrieved early to surround it. As a result, most IR-tree nodes are pruned early on and few edge comparison operations are involved.



(a) query time (b) IO cost

Figure 13: The effect of data set size

7. CONCLUSIONS

This paper reported a novel study on spatio-visual keyword queries to support WYRIWYS applications. Combining the visual angles and distances of spatial objects analytically, we proposed a quantitative visibility metric for objects with extents, relying on which we derived a hybrid ranking mechanism. To effectively process SVK queries, we proposed the COR method which consisted of two steps in the query processing and was capable of retrieving relevant objects incrementally. The experimental results confirmed the effectiveness and efficiency of our method when processing SVK queries.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation of China (Grant No. 60970124, 60903038, and 61170034).

9. REFERENCES

[1] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1):373–384, 2010.

[2] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD Conference*, pages 373–384, 2011.

[3] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD Conference*, pages 277–288, 2006.

[4] D. Cohen-Or, Y. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Trans. Vis. Comput. Graph.*, 9(3):412–431, 2003.

[5] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Cheong. *Computational Geometry: Algorithms and Applications*. Springer, second edition, 2000.

[7] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[8] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[9] Y. Gao, B. Zheng, G. Chen, W.-C. Lee, K. C. K. Lee, and Q. Li. Visible reverse k-nearest neighbor queries. In *ICDE*, pages 1203–1206, 2009.

[10] Y. Gao, B. Zheng, W.-C. Lee, and G. Chen. Continuous visible nearest neighbor queries. In *EDBT*, pages 144–155, 2009.

[11] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[12] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *SSDBM*, page 16, 2007.

[13] M. Kofler, M. Gervautz, and M. Gruber. R-trees for organizing and visualizing 3d gis databases. *Journal of Visualization and Computer Animation*, 11(3):129–143, 2000.

[14] K. C. K. Lee, W.-C. Lee, and H. V. Leong. Nearest surrounder queries. *IEEE Trans. Knowl. Data Eng.*, 22(10):1444–1458, 2010.

[15] S. Nutanong, E. Tanin, and R. Zhang. Incremental evaluation of visible nearest neighbor queries. *IEEE Trans. Knowl. Data Eng.*, 22(5):665–681, 2010.

[16] L. Shou, K. Chen, G. Chen, C. Zhang, Y. Ma, and X. Zhang. What-you-retrieve-is-what-you-see: a preliminary cyber-physical search engine. In *SIGIR*, pages 1273–1274, 2011.

[17] L. Shou, Z. Huang, and K.-L. Tan. Hdov-tree: The structure, the storage, the speed. In *ICDE*, pages 557–568, 2003.

[18] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou. Approximate string search in spatial databases. In *ICDE*, pages 545–556, 2010.

[19] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.