An Efficient Index Structure for Large-scale Geo-tagged Video Databases

Ying Lu Cyrus Shahabi Seon Ho Kim
Integrated Media Systems Center, University of Southern California, Los Angeles, California, USA
{ylu720,shahabi,seonkim}@usc.edu

ABSTRACT

An unprecedented number of user-generated videos (UGVs) are currently being collected by mobile devices, however, such unstructured data are very hard to index and search. Due to recent development, UGVs can be geo-tagged, e.g., GPS locations and compass directions, at the acquisition time at a very fine spatial granularity. Ideally, each video frame can be tagged by the spatial extent of its coverage area, termed Field-Of-View (FOV). In this paper, we focus on the challenges of spatial indexing and querying of FOVs in a large repository. Since FOVs contain both location and orientation information, and their distribution is non-uniform, conventional spatial indexes (e.g., R-tree, Grid) cannot index them efficiently. We propose a class of new R-tree-based index structures that effectively harness FOVs' camera locations, orientations and view-distances, in tandem, for both filtering and optimization. In addition, we present novel search strategies and algorithms for efficient range and directional queries on FOVs utilizing our indexes. Our experiments with a real-world dataset and a large synthetic video dataset (over 30 years worth of videos) demonstrate the scalability and efficiency of our proposed indexes and search algorithms and their superiority over the competitors.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation*

General Terms

Algorithms, Experimentation, Performance

Keywords

Geo-tag, index, scalability, user-generated video

1. INTRODUCTION

Driven by the advances in video technologies and mobile devices (e.g., smartphones), a large number of User-Generated-Videos (UGVs) are being produced and consumed by the public. According to Cisco [1], the overall mobile data traffic reached 1.5 exabytes per month in 2013 and it will reach 15.9 exabytes per month by 2018. Obviously, UGVs play a critical role in daily life, however, it is still challenging to organize and search such a huge amount of UGVs.

To overcome this challenge, we leverage smartphone sensors while capturing videos to model video content with its geospatial properties at the fine granularity level of frame (e.g., Field-Of-View [3]) . FOV model has been proven to be very useful for various media applications such as demonstrated by the online mobile media management system, MediaQ [6]. In the presence of such geo-metadata, we propose a new efficient index for a large scale geo-tagged video database.

SIGSPATIAL'14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2666310.2666480.

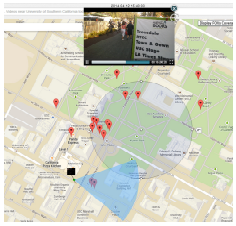


Figure 1: Range query in MediaQ [6]

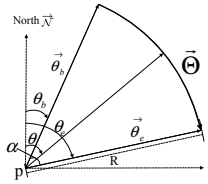


Figure 2: FOV model

Unlike conventional spatial objects (e.g., points, rectangles), FOV is a spatial object with orientation. For example, Fig. 1 shows the FOV (shaped in blue pie) of the video frame currently being dispalyed on MediaQ. Using FOVs, there are two typical queries on geo-tagged videos [8]: range and directional queries. A range query finds FOVs that overlap with a user-specified circular query area as shown in Fig. 1. A directional query finds FOVs whose orientations overlap with the user-specified direction within a range.

Note that "direction" discussed in this paper is an inherent attribute of FOVs. This is different than how direction has been treated in the past in the spatial database field, where they focused on directional relationships [9], direction-aware queries [7] (direction is *only* a component of a query), and moving directions. To distinguish from these "directions", we will use the term *"orientation"* when referring to the direction attribute of FOVs.

FOVs are spatial objects with both locations and orientations. Existing indexes (e.g., R-tree [4], Grid [8]) cannot efficiently support this type of data (see Sec. 2.2).

To overcome the drawbacks of the existing approaches, we propose a class of new index structures using both location and orientation information, termed *OR-trees*, building on the premises of R-tree. Our first straightforward approach uses R-tree to only index the camera locations of FOVs as *points* and then augments the index nodes to store their orientations. This variation of OR-tree is expected to generate smaller MBRs and reduce their dead spaces while supporting orientation filtering. To enhance further, we devise a second variation by adding an optimization technique in utilizing orientation information during node split and merge operations. Finally, in our third and last variation, we add the FOVs' viewable distances into the consideration during both filtering and optimization process.

Our experiments with both a real-world dataset and a large synthetic dataset (over 30 years worth of videos) demonstrate the scalability and efficiency of our proposed indexes and search algorithms and their superiority over the competitors.

2. PRELIMINARIES

2.1 Video Spatial Model and Query Definitions

We represent videos as a sequence of video frames, and each video frame is modeled as a Filed Of View (FOV) [3] as shown in Fig. 2. An FOV f is denoted as (p, R, $\overrightarrow{\Theta}$), in which, p is the camera location, R is the visible distance, $\overrightarrow{\Theta}$ is the orientation of the FOV in form of a tuple $<\overrightarrow{\theta_b}\ \overrightarrow{\theta_e}>$, where, $\overrightarrow{\theta_b}$ and $\overrightarrow{\theta_e}$ are the beginning and ending view directions in clockwise direction, the values of which are presented as $<\theta_b, \theta_e>$ w.r.t. the North $\overrightarrow{\mathcal{N}}$. During video recording using sensor-enable camera devices, we can obtain the camera view direction w.r.t the north θ and the visible angle α automatically [3]. Then we can derive $\theta_b = (\theta - \frac{\alpha}{2} + 360) mod 360$, and $\theta_e = (\theta + \frac{\alpha}{2} + 360) mod 360$.

We represent the coverage of a video v as a series of FOVs F_v. A video database $\mathcal{V}$ is represented as an FOV database $\mathcal{F} = \{F_{v_i} | \forall v_i \in \mathcal{V}\}$. Then, the problem of video search is transformed into spatial query on FOVs. Two typical geo-video queries, range and directional, are formally defined bellow.

$$RangeQ(Q_r, \mathcal{F}) \overset{def}{\Longleftrightarrow} \{f \in \mathcal{F} | f \cap Q_r \neq \emptyset\}$$

$$DirQ(Q_d, Q_r, \mathcal{F}) \overset{def}{\Longleftrightarrow} \{f \in \mathcal{F} | f.\vec{\Theta} \cap Q_d \neq \emptyset \ \wedge \ f \cap Q_r \neq \emptyset\}$$

2.2 Baseline Methods

2.2.1 R-tree

One baseline for indexing FOVs is using R-tree[4], which indexes the FOVs based on the MBRs of the visible scene of the FOVs. Consider the FOV objects in Fig. 3. Since R-tree is based on the optimization of minimizing the area of MBRs of FOVs, the MBRs of the leaf nodes of the R-tree are the dashed rectangles in Fig. 3 (assume the fanout is 2).

Range and Directional queries based on R-tree For the range query Q_r in Fig. 3, we need to access all the R-tree nodes ($R_1 \sim R_7$) since all of their MBRs overlaps Q_r. However, of which, only two FOVs f_1 and f_2 are results. For the directional query with the query direction interval Q_d ($0° - 90°$) and the query range Q_r, we also need to access all of the R-tree nodes since this R-tree cannot support orientation filtering.

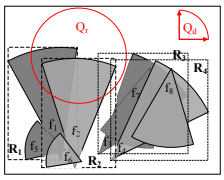
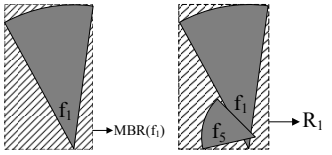


Figure 3: Sample dataset of FOVs



(a) FOV object (b) Index node

Figure 4: Dead spaces of object and R-tree node. Dashed area is the dead space.

Hence, R-tree has the following drawbacks for indexing FOVs. *Dead space*: Fig. 4 illustrates the "**dead spaces**" (or empty area, the area that is covered by the MBR of an R-tree node, but does not overlap with any objects in the subtree of the node [4]) of FOV f_1 and R-tree node R_1 in Fig. 3. Dead spaces will cause false positives for range queries, and thus increase both index node accesses and CPU computation cost. Taking the range query in Fig. 3 as an example, due to the dead spaces of index nodes R_3 and R_4, it needs to access R_3 and R_4, which are not necessary to be accessed since FOVs in neither R_3 nor FOVs in R_4 are results. *Large MBRs*: The area of the MBR of an R-tree node would be large due to the large visible scenes of the FOV objects enclosed in the node. With R-tree, large MBRs will increase the number of accessed node for a given range query since the decision whether to visit a node depends on if the MBR overlaps the query area [4]. *No orientation filtering*: With regular R-tree, there is no orientation information in the index nodes of the R-tree. *No orientation optimization*: R-tree is constructed based on the optimization of minimizing the covering area of FOV objects, without considering their directions.

2.2.2 Grid-based Index

Another approach that considers the directions of FOVs is Grid-based Index, termed as Grid [8], a three-level grid-based index structure based on viewable scene, camera locations and view directions. The first level indexes FOVs in a coarse grid, where each grid cell maintains the FOVs that overlap with the cell. At the second level, each first-level cell is divided into a set of subcells, each maintaining the FOVs whose camera locations are inside the cell. At the third level, it divides $360°$ into x intervals. Each direction interval maintains a list of FOVs whose orientations overlap with the interval. Grid uses the first and second levels for range filtering to process range queries and use the third level for orientation filtering to process directional queries.

However, Grid has the following drawbacks. First, it stores the location and orientation information at different levels, which is not efficient since video queries usually involve both location and orientation information of FOVs at the same time during query processing. Second, it is not suitable for indexing FOVs with different zoom levels and camera lens' properties since those FOVs have different viewable distances [3] and it will result in a large number of candidate second-level cells. Third, it performs poorly for skewed distribution of FOVs since the bucket occupancy of grid files rises very steeply for skewed distribution [5].

3. THE CLASS OF OR-TREES

To overcome the drawbacks of R-tree and Grid, we devise a class of new index structures combining camera locations, orientations and viewable distances of videos.

3.1 Orientation Augmented R-tree: OAR-tree

Recall that with R-tree, using MBRs to estimate FOVs will result in large MBRs, large "dead spaces" and the loss of orientation information. In this section, we introduce a new index called Orientation Augmented R-tree (OAR-tree) based on smaller MBRs, reduced "dead spaces", and incorporating orientation information in the index nodes, to accelerate the query efficiency.

In particular, for the leaf index nodes of an OAR-tree, instead of the MBRs of FOV objects, we store three values and a pointer to the actual FOV objects. Based on which, we can avoid the "dead spaces" of FOV objects to reduce false positives. Specifically, each leaf index node N of an OAR-tree contains a set of entries in the form of (Oid, p, R, $\overrightarrow{\Theta}$), where, as discussed in Sec. 2.1, Oid is the pointer to an FOV in the database; p is the camera location of the FOV object; R is its visible distance; and $\overrightarrow{\Theta}$ is its view orientation.

For internal index nodes, we replace 1) Oid with a pointer to the child node, 2) p with the MBR of all camera points in the child node, 3) R with an aggregate value representing all visible distances in the child node, and 4) $\overrightarrow{\Theta}$ with an aggregate value representing all orientations in the child node. Specifically, each non-leaf index node N of an OAR-tree contains a set of entries in the form of (Ptr, $MBRp$, $MinMaxR$, $\overrightarrow{MBO}$), where

- Ptr is the pointer to a child node of N;
- $MBRp$ is the MBR of the camera locations of the FOVs in the subtree rooted at Ptr; Obviously, $MBRp$ is much smaller than the MBR of FOVs in R-tree.
- $MinMaxR$ is a tuple <$MinR$, $MaxR$>, where $MinR$ (resp. $MaxR$) is the minimum (resp. maximum) visible distance of the FOVs in the subtree rooted at Ptr;

- $\overrightarrow{MBO}$ is the *Minimum Bounding Orientation* (MBO), defined in Definition 1 below, of the orientations of the FOVs in the subtree rooted at Ptr.

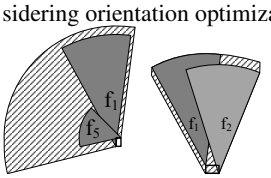
DEFINITION 1 (MINIMUM BOUNDING ORIENTATION). *Given a set of FOVs' orientations* $\Omega = \{\Theta_i < \theta_{bi}, \theta_{ei} >\}$, $1 \leq i \leq n$, *n is the number of orientations in* Ω, *then the Minimum Bounding Orientation (MBO) of* Ω *is the minimum angle in clockwise direction that covers all the orientations in* Ω, *i.e.,* $MBO(\Omega) =< \theta_b, \theta_e >$, *such that* $\overset{\frown}{\overrightarrow{\theta_b}\overrightarrow{\theta_e}} = \min_{\theta_{bi} \in \Omega} \{\max_{\theta_{ej} \in \Omega} \overset{\frown}{\overrightarrow{\theta_{bi}}\overrightarrow{\theta_{ej}}}\}$.

The OAR-tree stores the MBRs of camera locations, and incorporates the aggregate orientation and viewable distance information of all the child nodes to achieve smaller MBRs and orientation filtering. However, the OAR-tree is only based on the optimization of minimizing the covering area of the camera locations, which may result in large false positives for both range and directional queries. Similarly for the "dead space" of an R-tree node, we define the "Virtual Dead Space" of an OAR-tree node in Definition 2. Different from the dead space of an R-tree node where the coverage of an R-tree node, i.e., MBR, is stored, for the virtual dead space of an OAR-tree, its virtual coverage is not stored. However, both of them will produce false positives for range queries. Fig. 5(a) shows the virtual dead spaces of the OAR-tree node containing f_1 and f_5, and the OAR-tree node containing f_1 and f_2.

DEFINITION 2 (VIRTUAL DEAD SPACE). *Given an OAR-tree node* $N(MBRp, \overrightarrow{MBO}, MaxMinR)$, *then the* ***virtual dead space*** *of* N *is the area that is virtually covered by* N, *but does not overlap with any FOVs in the subtree of* N. *The* ***virtual coverage*** *of* N *is a convex such that any point in which can be covered by any FOV (p,* $\overrightarrow{\Theta}$, R), $\forall p \in N.MBRp$, $\forall \overrightarrow{\Theta} \in N.\overrightarrow{MBO}$, $\forall R \in N.MaxMinR$.

Consider Fig. 5(a) again, for the example in Fig. 3, FOV f_1 is grouped together with f_5 in the OAR-tree based on the camera point optimization. However, if f_1 is grouped together with f_2, additionally considering orientation information, then the virtual dead spaces of the OAR-tree node containing FOVs f_1 and f_5 will be significantly reduced and so does the number of false positives.

Based on this, we next discuss how to enhance OAR-tree by considering orientation optimization during the index construction.



(a) Location only (b) Location and orientation

Figure 5: Virtual dead spaces of OAR-tree nodes based on different optimizations.

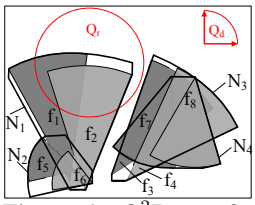


Figure 6: O²R-tree for the example in Fig.2.2.1

3.2 Orientation Optimized R-tree: O²R-tree

In this section, we propose a new index called Orientation Optimized R-tree (O²R-tree) that optimizes based on both the camera locations covering area and the similarity in orientation.

The stored information in O²R-tree index nodes is the same as that in the OAR-tree. The main difference between O²R-tree and OAR-tree is in the optimization criteria during the merging and splitting of the index nodes. We use the standard Quadratic Split algorithm [4] based on our proposed `Waste` function.

Given an O²R-tree entry $E(MBRp, MaxMinR, MBO)$ and an FOV object $f(p, R, \Theta)$, the covering area waste $\Delta Area(E, f)$ for the camera location is defined in Eqn(1).

$$\Delta Area(E, f) = Area(MBR(E, f)) - Area(E) \quad (1)$$

where $Area(MBR(E, f))$ is the area of the MBR of $E.MBRp$ and $f.p$; $Area(E)$ is the areas of $E_i.MBRp$. The angle waste for the view orientation is computed by Eqn(2)

$$\Delta Angle(E, f) = \overset{\frown}{\mathsf{MBO}(E.MBO, f.\Theta)} - \overset{\frown}{\overrightarrow{E.MBO}} - \overrightarrow{f.\Theta} \quad (2)$$

where $\overset{\frown}{\mathsf{MBO}(E.MBO, f.\Theta)}$ is the MBO of $E.MBO$ and $f.\Theta$.

Combining Eqn(1) and Eqn(2) using linear regression and normalization, we can compute the overall waste cost in Eqn(3).

$$Waste_{lo}(E, f) = \beta_l \frac{\Delta Area(E, f)}{max\Delta Area} + \beta_o \frac{\Delta Angle(E, f)}{max\Delta Angle} \quad (3)$$

In Eqn(3), $max\Delta Area$ (resp. $max\Delta Angle$) is the maximum of $\Delta Area(E, f)$ (resp. $\Delta Angle(E, f)$) for all the pair entries E_i and E_j to normalize the camera location (resp. orientation) waste. Parameters β_l and β_o, $0 \leq \beta_l, \beta_o \leq 1$, $\beta_l + \beta_o = 1$, are used to strike a balance between the area and angle wastes.

Fig. 6 gives the O²R-tree of our running example. It can prune O²R-tree node N_2 for both range and directional queries, instead of accessing all the index nodes as the OAR-tree.

3.3 View Distance and Orientation Optimized R-tree: DO²R-tree

Considering the camera location and orientation for optimization may still be insufficient. Hence, we discuss how to construct the index based on the optimization criterion incorporating the view distance information of FOVs, and we call the new index View Distance and Orientation Optimized R-tree (DO²R-tree).

The difference between DO²R-tree and O²R-tree is the optimization criteria. In the waste function of DO²R-tree in Eqn(5), it incorporates the view distance differences as given in Eqn(4).

$$\Delta Diff(E, f) = Diff(Ef) - Diff(E) \quad (4)$$

where $Diff(E)$ is the difference between maximum and minimum viewable distances of entry E. $Diff(Ef)$ is the difference between maximum and minimum viewable distances of node enclosing the viewable distances of E and f.

Combining all the wastes together, we can compute the overall waste cost in Eqn(5).

$$\begin{aligned} Waste_{lod}(E, f) &= \beta_l \frac{\Delta Area(E, f)}{max\Delta Area} + \beta_o \frac{\Delta Angle(E, f)}{max\Delta Angle} \\ &+ \beta_d \frac{\Delta Diff(E, f)}{max\Delta Diff} \end{aligned} \quad (5)$$

In Eqn(5), $max\Delta Diff$ is the maximum of $\Delta Diff(E, f)$ for all the pair entries E_i and E_j to normalize the visible distance. Parameters β_l, β_o and β_d, $0 \leq \beta_l, \beta_o, \beta_d \leq 1$, $\beta_l + \beta_o + \beta_d = 1$, are used to tune the impact of the three wastes. In particular, if $\beta_d = 0$, then DO²R-tree reduces to O²R-tree, and if also $\beta_o = 0$, then it becomes OAR-tree.

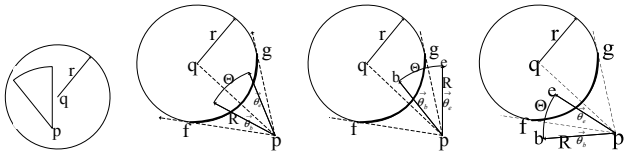
4. QUERY PROCESSINGS

We proceed to present the query algorithms based on DO²R-tree which is the generalization of the three indexes discussed in Sec. 3.

4.1 Range queries

At the high-level, the algorithm descends the DO²R-tree in the branch-and-bound manner, progressively checking whether each visited FOV object/index node overlap with the range query object. Subsequently, the algorithm decides whether to prune an object/index node, or to report the FOV object/index node (all the

FOVs in the index node) to be result(s). In the following, we will first present an exact approach to identify whether an FOV overlaps with the range query object, and then we exploit it to identify whether an index node should be accessed or not through two newly defined strategies: 1) pruning strategy and 2) total hit strategy.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

Figure 7: Overlap identifying for an FOV object

As shown in Fig. 7, there are four overlapping cases. Then we can derive the lemma bellow to identify whether an FOV is a result.

LEMMA 1 (OVERLAP IDENTIFYING FOR AN OBJECT). *Given an FOV $f\big(p, R, \Theta{<}\theta_b, \theta_e{>}\big)$ and a range query $Q_r(q, r)$, f overlaps with Q_r iff it satisfies Eqn(6), or Eqn(7), or Eqn(8), or Eqn(9).*

$$|pq| \leq r \tag{6}$$

$$|pq| \leq r + R \text{ and } \widehat{\overrightarrow{\theta_b}\overrightarrow{pq}} + \widehat{\overrightarrow{pq}\overrightarrow{\theta_e}} = \widehat{\overrightarrow{\Theta}} \tag{7}$$

$$|pq|\cos\widehat{\overrightarrow{pq}\overrightarrow{\theta_b}} - \sqrt{r^2 - (|pq|\sin\widehat{\overrightarrow{pq}\overrightarrow{\theta_b}})^2} \leq R \text{ and } \widehat{\overrightarrow{\theta_b}\overrightarrow{pq}} + \widehat{\overrightarrow{pq}\overrightarrow{\theta_e}} \neq \widehat{\overrightarrow{\Theta}} \tag{8}$$

$$|pq|\cos\widehat{\overrightarrow{pq}\overrightarrow{\theta_e}} - \sqrt{r^2 - (|pq|\sin\widehat{\overrightarrow{pq}\overrightarrow{\theta_e}})^2} \leq R \text{ and } \widehat{\overrightarrow{\theta_b}\overrightarrow{pq}} + \widehat{\overrightarrow{pq}\overrightarrow{\theta_e}} \neq \widehat{\overrightarrow{\Theta}} \tag{9}$$

Based on Lemma 1, we can develop the pruning strategy to examine if an index node $N\big(MBRp, {<}MinR, MaxR{>}, MBO{<}\theta_b, \theta_e{>}\big)$ can be pruned or not.

LEMMA 2 (PRUNING STRATEGY). *Index node N can be pruned if it satisfies Eqn(10), or Eqn(11), or Eqn(12),*

$$MinD(q, MBRp) \geq r + MaxR \tag{10}$$

$$MinA(MBO, MBRp, q) \geq \arcsin\frac{r}{MinD(MBRp, q)} \tag{11}$$

$$MinD(q, MBRp)\cos(MaxA(MBO, MBRp, q)) - \sqrt{r^2 - MinD^2(MBRp, q)\sin^2(MinA(MBO, MBRp, q))} \leq MaxR, \tag{12}$$

where $MinD(MBRp, q)$ is the minimum distance from q to $MBRp$

We next discuss the novel total hit strategy. We call an index node N a "total hit" iff all the objects in N overlap with the query circle. This is a new concept that does not exist with regular R-trees. If an index node N is a "total hit", then it is not necessary to exhaustively check for all the FOVs in N one by one, so the processing cost can be significantly reduced.

LEMMA 3 (TOTAL HIT STRATEGY). *All the FOVs in the subtree of N can be reported as results if it satisfies Eqn(13), or all the equations (14), (15) and (16),*

$$MaxD(q, MBRp) \leq r \tag{13}$$

$$MaxD(q, MBRp) \leq r + MinR \tag{14}$$

$$MaxA(MBO, MBRp, q) \leq \arcsin\frac{r}{MaxD(MBRp, q)} \tag{15}$$

$$MaxD(q, MBRp)\cos(MinA(MBO, MBRp, q)) - \sqrt{r^2 - MaxD^2(MBRp, q)\sin^2(MaxA(MBO, MBRp, q))} \leq MinR \tag{16}$$

Due to the space limitation, we included the proofs of Lemmas 1, 2, and 3 in our technical report [2].

4.2 Directional queries

Given a direction interval Q_d, we can easily decide whether the orientation of a DO^{2}R-tree node overlaps with Q_d. The directional query algorithm also follows a branch-and-bound process, progressively applying the search strategies. Note that we apply the range search strategies and the orientation filtering at the same time to decide if a DO^{2}R-tree node can be pruned or is a "total hit".

5. PERFORMANCE EVALUATION

We conducted experimental studies to evaluate the efficiency of our proposed indexes and search algorithms: OAR-tree, O^{2}R-tree, and DO^{2}R-tree for range and directional queries using a real-world dataset and big synthetically generated datasets (more than 30 years worth of videos). We observed that both O^{2}R-tree and DO^{2}R-tree significantly outperformed the baseline indexes (i.e., R-tree and Grid) for both range and directional queries. Specifically, O^{2}R-tree (resp., DO^{2}R-tree) accessed around 40% (resp., 50%) less pages than Grid, and around 50% (resp., 60%) less than R-tree for range queries. In additional, O^{2}R-tree (resp., DO^{2}R-tree) accessed about 70% (resp., 65%) less number of pages than Grid and accessed about 67% (resp., 63%) less than R-tree for directional queries. This demonstrates that the orientation optimization in building O^{2}R-tree and DO^{2}R-tree was more effective in supporting directional queries. Another observation is that OAR-tree outperformed Grid slightly for directional queries and even incurred a slightly more page accesses than R-tree for range queries. The results demonstrated that not the simple consideration of orientation but the optimization criteria considering the orientation significantly facilitated the reduction of the dead spaces of tree nodes and subsequently leading to the reduction of false positives.

6. CONCLUSION AND FUTURE WORK

We represented video data as a series of spatial objects with orientations, i.e., FOVs, and proposed a class of R-tree-based indexes that can index location, orientation, and distance information of FOVs for both filtering and optimization. Further, two novel search strategies were proposed for fast video range and directional queries on top of our index structures. The experimental results demonstrate the superiority of our indexes comparing to conventional ones. We intend to extend this work in two directions: 1) to extend our indexes to the cloud for even larger sets of video data, 2) to study the insertion and update costs of our indexes for batch insertion of video.

Acknowledgments

This research has been funded in part by NSF grant IIS-1320149, the USC IMSC, and unrestricted cash gifts from Google and Northrop Grumman. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

7. REFERENCES

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf.

[2] An Efficient Index Structure for Large-scale Geo-tagged Video Databases. http://www.cs.usc.edu/research/technical-reports-list.htm?#2014.

[3] A. S. Ay, R. Zimmermann, and S. H. Kim. Viewable Scene Modeling for Geospatial Video Search. In *ACM Intl. Conf. on MM*, pages 309–318, 2008.

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

[5] V. Jain and B. Shneiderman. Data structures for dynamic queries: An analytical and experimental evaluation. In *Proc. of the Workshop on Advanced Visual Interfaces. NY: ACM*, pages 1–11, 1994.

[6] S. H. Kim, Y. Lu, G. Constantinou, C. Shahabi, G. Wang, and R. Zimmermann. Mediaq: Mobile multimedia management system. In *ACM MMSys*, pages 224–235, 2014.

[7] X. Liu, S. Shekhar, and S. Chawla. Object-based directional query processing in spatial databases. *Proc. of IEEE TKDE*, 15(2):295–304, Feb. 2003.

[8] H. Ma, S. A. Ay, R. Zimmermann, and S. H. Kim. Large-scale geo-tagged video indexing and queries. *GeoInformatica*, Dec. 2013.

[9] Y. Theodoridis, D. Papadias, and E. Stefanakis. Supporting direction relations in spatial database systems. In *Proc. of the 7th Intl. Symposium on Spatial Data Handling(SDH'96)*, 1996.