Viewable Scene Modeling for Geospatial Video Search

Sakire Arslan Ay
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
arslan@usc.edu

Roger Zimmermann
School of Computing
National University of Singapore
Singapore 117590
rogerz@comp.nus.edu.sg

Seon Ho Kim
Department of Computer Science
University of Denver
Denver, CO 80208
seonkim@cs.du.edu

ABSTRACT

Video sensors are becoming ubiquitous and the volume of captured video material is very large. Therefore, tools for searching video databases are indispensable. Current techniques that extract features purely based on the visual signals of a video are struggling to achieve good results. By considering video related meta-information, more relevant and precisely delimited search results can be obtained. In this study we propose a novel approach for querying videos based on the notion that the geographical location of the captured scene in addition to the location of a camera can provide valuable information and may be used as a search criterion in many applications. This study provides an estimation model of the viewable area of a scene for indexing and searching and reports on a prototype implementation. Among our objectives is to stimulate a discussion of these topics in the research community as information fusion of different georeferenced data sources is becoming increasingly important. Initial results illustrate the feasibility of the proposed approach.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query processing*; H.2.4 [**Database Management**]: Systems—*Multimedia databases*; C.4 [**Performance of Systems**]: Modeling techniques

General Terms

Algorithms, Measurement, Performance

Keywords

Video search, georeferencing, meta-data, GPS

1. INTRODUCTION

Due to technological advances, an increasing number of video clips are being collected from various devices and stored for a variety of purposes such as surveillance, monitoring, reporting, or entertainment. These acquired video clips contain a tremendous amount of visual and contextual information that makes them unlike any other media type. However, even now, there are no effective ways to index and search video data at the high semantic level preferred by humans. Text annotations of video can be utilized for search, but high-level concepts must often be added by hand and hence this manual task is laborious and cumbersome for large video collections. Content based video retrieval is in its infancy, very challenging and still not always satisfactory.

Some types of video data are naturally tied to geographical locations. For example, video data from traffic monitoring may not have any meaning without its associated position information. Thus, in such applications, one needs a specific location to retrieve the traffic video at that point or in that region. Hence, combining video data with its location coordinates can provide an effective way to index and search videos, especially when a database handles an extensive amount of video data. Note that location information can be collected by various small devices attached to a camera, such as a global positioning system (GPS) sensor (see Figure 1). A preliminary example of this type of work is Google Earth, which implements such a concept with panoramic images, but only from a high elevation (sky view). A user can find a top view of a point or region given a query point. The current implementation is innovative but has some limitations.

We believe that *georeferenced video search* will play a prominent role in many future applications. However, there are still many open, fundamental research questions in this field. Most videos captured are not panoramic and as a result the *viewing direction* becomes very important. GPS data only identifies object locations and therefore it is imperative to investigate the natural concepts of a viewing direction and a view point. For example, we may be interested to view a building only from a specific angle. The question arises whether a video database search can accommodate such human friendly views. The collection and fusion of multiple sensor streams such as the camera location, field-of-view, direction, etc., can provide a comprehensive model of the *viewable scene*. The objective then is to index the video data based on the human viewable space and therefore to enable the retrieval of more meaningful and recognizable scene results for user queries. Cameras may also be mobile and thus the concept of a camera location is extended to a trajectory. Consequently, finding relevant video segments

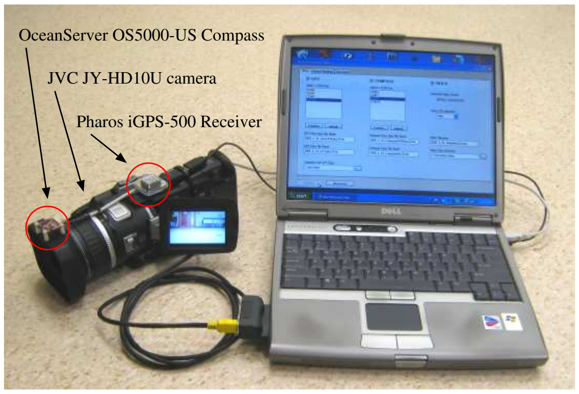


Figure 1: Experimental hardware and software to acquire georeferenced video.

becomes very challenging. In this study we propose a general methodology to address these and related issues.

One example application that would benefit from our georeferenced video search framework is *geospatial decision making*. The recent rapid increase in the amount of geospatial data available has motivated efforts to integrate multiple geospatial data sets for the purpose of extracting useful information and assisting decision makers. Event extraction – while difficult in the visual signal domain – shows promising results from geospatial information integration and data fusion. Our study provides the following contributions.

- **Automatic annotation of video clips with the camera viewing direction.** While the concept of meta-data annotation has been investigated before, we believe our method is the first to consider the viewing direction.
- **Modeling of the viewable scene.** We propose a viewable scene model that strikes a balance between the complexity of its analytical description and the efficiency with which it can be used for fast searches.
- **Prototype feasibility study.** We have implemented a prototype to demonstrate the feasibility of acquiring, storing, searching, and retrieving video based on our approach.
- **Demonstration of benefits.** From our implementation results we are able to illustrate the benefits of our approach in retrieving the most relevant video segments for a given query.

Before elaborating on our approach in detail, Section 2 contains a brief discussion and survey of related work. We describe the proposed approach for georeferenced video search in Section 3. This is followed by a presentation of results based on a real-world data set in Section 4. Finally, Section 5 discusses open issues and future research directions.

2. RELATED WORK

Associating GPS coordinates with digital media (images and videos) has become quite popular [15]. We will start our survey with methods that specifically consider still images and then move on to videos. Lastly we will describe the prior work in the area of indexing and storage.

Techniques for Images. There has been significant research on organizing and browsing personal photos according to location and time. Toyama et al. [19] introduced a meta-data powered image search and built a database, also known as World Wide Media eXchange (WWMX), which indexes photographs using location coordinates (latitude/longitude) and time. This work specifically explores methods for acquiring location tags, optimizing an image database for efficient geo-tagged image search and exploiting meta-data in a graphical user interface for browsing. A number of additional techniques in this direction have been proposed [13, 14]. There are also several commercial web sites [2, 3, 4] that allow the upload and navigation of geo-refenced photos. All these techniques use camera geo-coordinates as the reference location in describing images. Recently Ephstein et al. [5] proposed organizing large collections of images based on scene semantics. The authors related images with their view frustum (viewable scene) and used a scene-centric ranking (termed Geo-Relevance) to generate a hierarchical organization of images. Although this work is similar to ours in using the camera field-of-view to describe the viewable scene, its main contribution is for image classification, completely ignoring the time dimension. Our work describes a more broad scenario that considers mobile cameras capturing geo-tagged videos and the associated view frustum, which is dynamically changing over time. Several more methods are based on image annotations and a camera's viewable scene [18] and organizing geo-tagged images in 3D space using camera properties [10].

Techniques for Video. There exist only a few systems that associate videos with their corresponding geo-location. Hwang et al. and Kim et al. propose a mapping between the 3D world and the videos by linking the objects to the video frames in which they appear [9, 11]. Their work mentions using GPS location and camera direction to build links between video frames and world objects. However they neglect to provide any details on how this is established and how accurate it is. Furthermore, their work differs from ours by only targeting specific objects within the geo-space – ignoring the time information – whereas our approach keeps track of the viewable region over time. More closely related to our work, Liu et al. [12] presented a sensor enhanced video annotation system (referred to as SEVA) which enables searching videos for the appearance of particular objects. SEVA serves as a good example to show how a sensor rich, controlled environment can support interesting applications, however it does not propose a broadly applicable approach to geospatially annotate videos for effective video search. For example, in the context of what is being annotated, SEVA tags video streams with the information about what was visible within the video, whereas we target to annotate the viewable region within the video. There are number of research projects experimenting with location-aware interactions which try to geo-spatially describe what a human can see in the real world, rather than what a camera may record. Simon et al. [16] describe a web based application framework for retrieving relevant World Wide Web documents for the objects within the user's viewable scene in the real world. We believe that our approach, when enhanced with an efficient spatio-temporal storage and indexing mechanism, will serve as a general purpose and flexible video search mechanism that is applicable to any types of video with associated location and direction tags. Consequently it can be the basis for a tremendous number of multimedia applications.

Indexing and Storage. There is a strong need for an indexing structure for efficient storage and querying of our geo-referenced video annotations. Conventional database

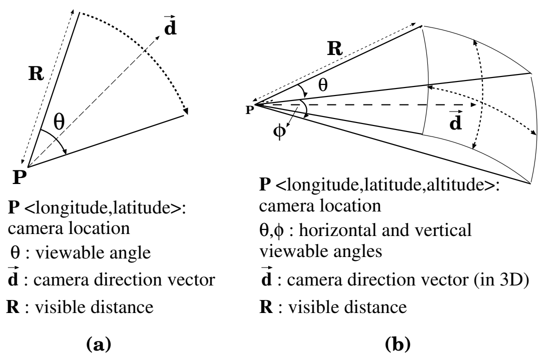

R
$\vec{d}$
θ
P
P <longitude,latitude>: camera location
θ : viewable angle
$\vec{d}$: camera direction vector
R : visible distance
(a)
R
θ
P
ϕ
$\vec{d}$
P <longitude,latitude,altitude>: camera location
θ,ϕ : horizontal and vertical viewable angles
$\vec{d}$: camera direction vector (in 3D)
R : visible distance
(b)

Figure 2: Illustration of *FOVScene* model (a) in 2D (b) in 3D.

and GIS technology can be used to partially manage some non-temporal properties of video objects such as the location of a camera and the trajectory of camera movement. The most important characteristics of geo-referenced video are the human recognizable and viewable space of scenes, human viewing directions, and management of the inherent uncertainty in visual information – all features that are completely lacking in conventional spatio-temporal databases. Some researchers have investigated the relative geographical locations of objects within a scene [17]. However, to the best of our knowledge, there has been no study in modelling the actual geographical information of video scenes and utilizing it for an efficient video search.

Commercial products. There exist several GPS-enabled digital cameras which can save the location information with the digital image file as a picture is taken (e.g., Sony GPS-CS1, Ricoh 500SE, Jobo Photo GPS). Very recent models additionally record the current heading (e.g., Ricoh SE-3, Solmeta DP-GPS N2). All current cameras support geotagging for still images only. We believe that, as the use of these cameras increases, more location and direction tagged videos will be produced and there will be a strong need to perform efficient and effective search on those video data.

3. APPROACH

This study focuses on how to quantify, store and query the viewable scene of captured videos. We model the viewable space of a scene with parameters such as the camera location, the angle of the view, and the camera direction. As the camera moves or rotates over time, the viewable scene changes as well. This dynamic scene information has to be acquired from sensor-equipped cameras, stored within an appropriate catalog or schema and indexed for efficient querying and retrieval. Our proposed approach consists of three components: 1) modeling of the viewable scene, 2) data acquisition, and 3) indexing and querying. We will now describe each of these in turn.

3.1 Modeling of Viewable Scene

A camera positioned at a given point P in geo-space captures a scene which we call the *viewable scene* of the camera. In the field of computer graphics, this area is referred to as *camera field-of-view* (FOV for short). We will use the terms 'viewable scene' and 'field of view (FOV)' interchangeably throughout this manuscript.

The meta-data related to the geo-properties of a camera and its captured scene are the following. (1) The camera position P is the ⟨latitude, longitude⟩ coordinates read from a positioning device (e.g., GPS). (2) The camera direction vector $\overrightarrow{d}$ is obtained based on the orientation angle provided by a digital compass. (3) The camera viewable angle θ describes the angular extent of the scene imaged by the camera. The angle θ is calculated based on the camera and lens properties for the current zoom level [6]. (4) The far visible distance R is the maximum distance at which a large object within the camera's field-of-view can be recognized. Based on the availability of the above data one can model the viewable scene of a camera in different ways. One simple and straightforward approach – using only the GPS location of a camera (P) – is to model the viewable scene as a point. We will refer to this scene description as *PointScene(P)* throughout the manuscript. In the presence of information about both the GPS location P and the visible distance R, the viewable scene can be modeled as a circle with radius R centered at P. This method covers all the possible area a camera might see when the view direction is not known. We will use the term *CircleScene(P, R)* to refer to this scene description. If all the meta-data (P, R, $\vec{d}$, θ) are available, one can model the viewable area more accurately, which results in more effective indexing and searching.

This paper proposes the *FOVScene* model, which describes a camera's viewable scene in 2D space by using the four parameters: camera location P, camera orientation vector $\vec{d}$, viewable angle θ and visible distance R (see Eqn. 1).

$$FOVScene(P, \vec{d}, \theta, R) \tag{1}$$

The full field-of-view is obtained with the maximum visual angle, which depends on the lens/image sensor combination used in the camera [8]. Smaller image sensors have smaller field-of-view than larger image sensors (when used with the same lens). Alternatively, shorter focal-length lenses have a larger field-of-view than longer focal-length lenses (when used with the same image sensor). The viewable angle θ can be obtained via the following formula (Eqn. 2) [8]:

$$\theta = 2tan^{-1}\frac{y}{2f} \tag{2}$$

where y is the size of the image sensor and f is the focal length of the lens. The relationship between the visible distance R and the viewable angle θ is given in Eqn. 3 [8]. As the camera view is zoomed in or out, the θ and R values will be adjusted accordingly.

$$\theta = 2\arctan\left(\frac{y(R\cos(\frac{\theta}{2}) - f)}{2fR\cos(\frac{\theta}{2})}\right) \tag{3}$$

We assume that the video content is captured from a sensor-equipped camera (see Figure 1), which can accurately estimate its current location P, orientation $\vec{d}$ and visual angle θ. In 2D space, the field-of-view of the camera at time t, ($FOVScene(P, \vec{d}, \theta, R, t)$) forms a pie-slice-shaped area as illustrated in Figure 2a. Figure 2b shows an example camera *FOVScene* volume in 3D space. For a 3D *FOVScene* representation we would need the altitude of the camera location point and the pitch and roll values to describe the camera heading on the zx and zy planes (i.e., whether camera is directed upwards or downwards). We believe that the extension to 3D is straightforward, especially since we already acquire the altitude level from the GPS device and the pitch and roll values from the compass. For our initial setup we will represent *FOVScene* in 2D space only. We plan to work on the 3D extension in our future work.

3.2 Meta-Data Acquisition

The viewable scene of a camera changes as it moves or changes its orientation in geo-space. In order to keep track of the *FOVScene* of a moving camera over time, we need to record its location (P), direction ($\vec{d}$) and viewable angle (θ) with a certain frequency and produce time stamped meta-data together with time stamped video streams. Our meta-data streams are analogous to sequences of $\langle P, \vec{d}, \theta, R, t \rangle$ quintuples, where t is the time instant at which *FOVScene* information is recorded. In large scale applications, there may be thousands of moving cameras with different sensing capabilities. Each camera will record its location and *FOVScene* description with a different sampling frequency. We do not make any assumptions about how frequently a camera should record its *FOVScene* coverage.

Recording georeferenced video streams. Our sensor rich video recording system incorporates three devices: a video camera, a 3D digital compass, and a Global Positioning System (GPS) device. We assume that the optical properties of the camera are known. The digital compass, mounted on the camera, periodically reports the direction in which the camera is pointing. The camera location is read from the GPS device as a ⟨latitude, longitude⟩ pair. Video can be captured with various camera models – we use a high-resolution (HD) camera. Our custom-written recording software receives direction and location updates from the GPS and compass devices as soon as new values are available and records the updates along with the current computer time and coordinated universal (UTC) time. Each recorded GPS update includes: (1) the current latitude and longitude, (2) the local time when the location update was received (in milliseconds), (3) the satellite time (in UTC) for the location update (in seconds), (4) the current speed over ground, and (5) the altitude. Each recorded compass update includes: (1) the current heading angle (with respect to the North), (2) the local time when the direction update was received (with millisecond granularity), and (3) the current pitch and roll values. Altitude, pitch and roll measurements are not directly used in the 2D camera viewable scene calculation, however they would be required to estimate a 3D viewable scene which defines a cone shaped volume (see Figure 2b). Furthermore, pitch and roll values are useful for analyzing the accuracy of the heading measurements.

Each video data packet received from the camera is processed in real time to extract frame timecodes. Extracted timecodes are recorded along with the local computer time when the frame was received. Since video data is received from the camera as data packet blocks, all frames within a video packet will initially have the same local timestamp. Timestamping accuracy can be improved by adjusting the local time backwards within the data block based on the frame rate. Creating a frame level time index for the video stream will minimize the synchronization errors that might occur due to clock skew between the camera clock and the computer clock. In addition, such a temporal video index, whose timing is compatible with other datasets, enables easy and accurate integration with the GPS and compass data. We also keep track of the size of the video data captured since the beginning of the capture and record the byte offset for each video frame.

Calculating viewable angle (θ) and visible distance (R). Assuming that the optical focal length f and the size of the camera image sensor y are known, the camera viewable angle θ can be calculated through Eqn. 2. The default focal length for the camera lens is obtained from the camera specifications. However, when there is a change in the camera zoom level, the focal length f and consequently the viewable angle θ will change. To capture the change in θ, the camera should be equipped with a special unit that will measure the focal length for different zoom levels. Such functionality is not commonly available in today's off-the-shelf digital cameras and camcorders. To simulate the changes in the viewable angle, we have manually recorded the exact video timecodes along with the change in the zoom level. Using the Camera Calibration Toolbox [1] we have measured the f value for five different zoom levels (from the minimal to the maximal zoom level). For all other zoom levels, the focal length f is estimated through interpolation. The visible distance R can be obtained based on the following equation (Eqn. 4),

$$R = \frac{fh}{y} \tag{4}$$

where f is the lens focal length, y is the image sensor height and h is the height of the target object that will be fully captured within a frame. With regard to the visibility of an object from the current camera position, the size of the object also affects the maximum camera-to-object viewing distance. For large objects (e.g., mountains, high buildings) the visibility distance will be large whereas for small objects of interest the visibility distance will be small. For simplicity in our initial setup we assume R to be the maximum visible distance for a fairly large object. Let us assume that we would like to capture a two story, approximately 8.5m-tall building at a certain distance from the camera. We claim that, assuming good lighting conditions and no obstructions, the building can be considered visible within a captured frame if it occupies at least $1/20^{th}$ of the full image height. Therefore the total target height we would like to capture with the camera is $h = 200$m. As an example, for our JVC JY-HD10U camera, $f = 5.2$mm and the CCD image sensor height $y = 3.6$mm, and therefore $R = 250$m. We currently target a mid-range far visible distance of 200-300m. We believe that this range best fits with typical applications that would most benefit from our georeferenced video search (e.g., traffic monitoring, surveillance). Close-up and far-distance will be considered as a part of our future research.

Timing and Synchronization. As described above, all meta-data entries (GPS, compass updates and video frame timecodes) have millisecond-granular timing. In addition to the local time, we record the satellite time (in UTC) that is received along with the GPS location update. The use of the recorded satellite time can be twofold: (1) it enables synchronizing the current computer time with the satellite time (2) it may be used as the time base when executing temporal queries, i.e., by applying the temporal condition of the query to the satellite time. Timestamping the camera viewable scenes with the satellite time ensures a global temporal consistency among all georeferenced video collections. All three meta-data streams need to be combined and stored as a single stream with an associated common time index. In a sensor-rich system with several attached devices, one challenge is how to synchronize the sensor data read from the attached devices which have different data output rates. Let f_i be the output rate for the device with ID i. Intuitively, the data frequency of the combined stream will be at most $\min(f_i), \forall i$. A naive way is to create the combined data stream during data collection and record the most up-to-date sensor readings as a single record every $1/\min(f_i)$ seconds using the local time as the common timestamp. This method can only consider past data updates from devices. However it is possible that a new update from a device is available immediately after a combined tuple has been generated. Therefore, the naive method does not guarantee that temporally closest values among all device updates will be matched in the combined stream. A better approach is to record separate data streams for each device, where each data entry is timestamped with the time when the update was received from the device. Later these data streams can be combined with a 2-pass algorithm. Such an algorithm processes data in a sliding time window centered at the current time. It will always match the data entries that have the closest timestamps (past or future). In our setup $f_{GPS} = 1$ sample/sec, $f_{compass} = 40$ samples/sec, and $f_{video} = 30$ samples/sec. Therefore we match each GPS entry with the temporally closest video frame timecode and compass direction. It is further possible to estimate missing data items in low frequency data streams by applying an interpolation technique. *Positional Interpolation* is widely used in location-based services to obtain the entire movement/trajectory of a moving object. In our data collection prototype, we match the temporally closest direction updates and frame timecodes and estimate the location at that timestamp through a positional interpolation technique. The ground speed of the camera point, which is available at GPS location updates, is useful for a better location estimation. Note that since $f_{video} < f_{compass}$ there is no need to interpolate compass heading values.

Measurement Errors. The accuracy of the *FOVScene* calculation is somewhat dependent on the precision of the location and heading measurements obtained from GPS and compass devices. A typical GPS device is accurate approximately within 10 meters. In our proposed viewable scene model, the area of the region that a typical HD camera captures (*FOVScene*) is on order of tens of thousands of square meters (e.g., at full zoom-out approx. 33,000m^2). Therefore, a difference of 10m is not very significant compared to the size of the viewable scene we consider. Additionally, missing GPS locations – due to various reasons such as a tunnel traversal – can be recovered through estimation such as interpolation. There exists extensive prior work on estimating moving object trajectories in the presence of missing GPS locations. An error in the compass heading may be more significant. Many digital compasses ensure azimuth accuracy of better than 1° (e.g., about 0.6° for the OS5000 digital compass in our system), which will have a minor effect on the viewable scene calculation. However, when mounted on real platforms the accuracy of a digital compass might be affected by local magnetic fields or materials. For our experiments the compass was calibrated within the setup environment to minimize any distortion in compass heading. It is also worth mentioning that multimedia applications often tolerate some minor errors. When a small object is at the edge of viewable scene but is not included in the modeled area, it might not be recognized by a human observer.

3.3 Indexing and Querying

The next task after collecting georeferenced meta-data is to semantically describe them so that accurate and efficient analysis on the camera viewable scenes is possible. An intuitive way is to store a separate *FOVScene* quintuple for each video frame. An example schema is given in Table 1. The *FOVScene* coverage of a moving camera over time is analogous to a moving region in the geo-spatial domain, therefore traditional spatio-temporal query types, such as range queries, k nearest neighbor (kNN) queries or spatial joins, can be applied to the *FOVScene* data. In our initial work, we limit our discussion to range queries. The typical task we would like to accomplish is to extract the video segments that capture a given area of interest. As explained in Section 3.2, for each video frame with timecode t, we keep track of the camera location coordinates P, the camera orientation $\vec{d}$, and the camera view angle θ. Therefore, we can construct the *FOVScene(t,P,$\vec{d}$,θ,R)* description for every video frame. Hence, for a given area of interest Q, we can extract the sequence of video frames whose viewable scene overlap with Q. Going from most specific to most general, the query region Q can be a point, a line (e.g., a road), a poly-line (e.g., a trajectory between two points), a circular area (e.g., neighborhood of a point of interest), a rectangular area (e.g., the space delimited with roads) or a polygon area (e.g., the space delimited by certain buildings, roads and other structures). In our initial setup, we simply return all frames with timecode t, for which *SceneIntersect(Q,FOVScene(t,P,$\vec{d}$,θ,R))* is true. Algorithm 1 formalizes the methodology for checking whether a particular *FOVScene* area intersects with the query region Q. Algorithm 1 calls the subroutines *pointFOVIntersect* (given in Algorithm 2) and *edgeFOVIntersect* (given in Algorithm 3). The computations for the other two scene models, *SceneIntersect(Q,CircleScene(t,P,R))* and *SceneIntersect(Q,PointScene(t,P)*, are trivial and omitted due to space limitations.

camera_id	ID of camera
video_id	ID of video
frame_timecode	Timecode for the current frame in video (extracted from video)
location	Current camera position (P) (read from GPS)
visual_angle	Angular extent for camera field-of-view (θ) (calculated based on camera properties)
direction	Current camera direction (d) (read from compass)

Table 1: Example schema for *FOVScene* representation.

Algorithm 1 $SceneIntersect(Q, FOVScene(t, P, \vec{d}, \theta, R))$

```
$Q \leftarrow$ Given convex polygon shaped query region
$Q =< V, E >$ ($V \leftarrow$Set of vertices in $Q$; $E \leftarrow$Set of edges in $Q$)
$FOVScene(t, P, \vec{d}, \theta, R) \leftarrow$ FOV region for frame $t$
$pointPolygonIntersect(P, Q) \leftarrow$ returns true if point $P$ is in $Q$

if $pointPolygonIntersect(P, Q)$ is $true$ then
  return $true$;
end if
for $\forall v \in V$ do
  if $pointFOVIntersect(v, FOV(t, P, \vec{d}, \theta, R)) == true$ then
    return $true$;
  end if
end for
for $\forall e \in E$ do
  if $edgeFOVIntersect(e, FOV(t, P, \vec{d}, \theta, R)) == true$ then
    return $true$;
  end if
  return $false$;
end for
```

One problem with such a representation on top of a relational model is the computational overhead. In a typical query and for a given time interval, all *FOVScene* quintuples

Algorithm 2 $pointFOVIntersect(q, FOVScene(t, P, \vec{d}, \theta, R))$

```
$q \leftarrow$ Given query point
$FOVScene(t, P, \vec{d}, \theta, R) \leftarrow$ FOV region for frame $t$
$EarthDistance(q, P) \leftarrow$ returns the earth distance between points $q$ and $P$
if $EarthDistance(q, P)) \leq R$ then
    $\alpha$=angle of $\vec{d}$ with respect to north
    $\delta$=angle of vector $\vec{Pq}$ with respect to north
    if $\alpha - (\frac{\theta}{2}) \leq \delta \leq \alpha + (\frac{\theta}{2})$ then
        return $true$;
    end if
end if
return $false$;
```

Algorithm 3 $edgeFOVIntersect(q, FOVScene(t, P, \vec{d}, \theta, R))$

```
$e \leftarrow$ Given edge
$FOVScene(t, P, \vec{d}, \theta, R) \leftarrow$ FOV region for frame $t$
$LinePointDistance(P, e) \leftarrow$ returns the shortest earth distance between point $P$ and closest point on line $e$
$< p_1, p_2 > = lineCircleIntersect(e, P, R)$; {return the two points that line $e$ intersects with the circle centered at $P$ with radius $R$. Note that $p_1 = p_2$ if $e$ is tangent to circle}
if $LinePointDistance(P, e)) \leq R$ then
    $\alpha$=angle of $\vec{d}$ with respect to north
    $\delta_1$=angle of vector $\vec{Pp_1}$ with respect to north
    $\delta_2$=angle of vector $\vec{Pp_2}$ with respect to north
    if $[\alpha - (\frac{\theta}{2}), \alpha + (\frac{\theta}{2})]$ overlaps with $[\delta_1, \delta_2]$ then
        return $true$;
    end if
end if
return $false$;
```

that belong to that time frame have to be checked for overlaps with the query area. A more appropriate approach is to define the *FOVScene* in the spatial domain with a pie-slice-shaped area as described in Section 3.1 and then estimate it with a Minimum Bounding Rectangle (MBR) (see Figure 3). An intuitive structure to index these MBRs is the R-tree [7] family, one of the most popular index structures for rectangles. We believe that there is a strong need for a better data structure to describe and index *FOVScene* descriptions. This is one of the challenges that will need to be explored further in the future.

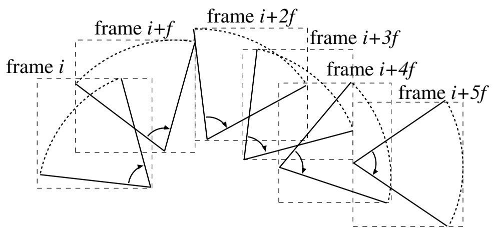


Figure 3: MBR estimations of camera FOVs.

In this study we restrict our example queries to simple spatiotemporal range searches. However, using the camera view direction ($\vec{d}$) in addition to the camera location (P) to describe the camera viewable scene provides a rich information base for answering more complex geospatial queries. For example, if the query asks for the views of an area from a particular angle, more meaningful scene results can be returned to the user. Alternatively, the query result set can be presented to user as distinct groups of resulting video sections such that videos in each group will capture the query region from a different view point. Similarly, when query results are presented to the user, the video segments can be ranked based on how relevant they are to the query requirements and user interests. There are several factors that need to be taken into account in estimating relevance. Among them are the amount of overlap, the closeness of the overlapping area to the camera location, the angle at which the camera is viewing the overlap area, etc. Furthermore, the way *FOVScene* overlaps with the query area may indicate the importance of the video clip. Some further aspects of a complete system to query georeferenced videos – such as indexing and query optimization – will be explored as part of our future work.

4. EXPERIMENTAL EVALUATION

4.1 Data Collection and Methodology

To collect georeferenced video data, we have constructed a prototype system which includes a high-resolution (HD) camera, a 3D compass and a GPS receiver. We used the JVC JY-HD10U camera with a frame size of approximately one megapixel (1280x720 pixels at a data rate of 30 frames per second). It produces MPEG-2 HD video streams at a rate of slightly more than 20 Mb/s and video output is available in real time from the built-in FireWire (IEEE 1394) port. To obtain the orientation of the camera, we employed the OS5000-US Solid State Tilt Compensated 3 Axis Digital Compass, which provides very precise tilt compensated heading, roll and pitch information. The compass delivers an output rate of up to 40 samples per second. To acquire the camera location we used the Pharos iGPS-500 GPS receiver. We have built a software to acquire, process, and record the georeferences along with the MPEG2 HD video streams. MPEG2 video is processed in real-time (without decoding the stream) and each video frame is associated with its viewable scene information. More details on acquisition and synchronization issues have been provided in Section 3.2. Although our sensor rich video recording system has been tested mainly with cameras that produce MPEG-2 video output, with little effort it can be configured to support any digital camera producing compressed or uncompressed video streams. Both progressive and interlaced video can be processed. With interlaced video the camera location and heading is recorded for each frame consisting of an odd and even field pair.

Figure 1 shows the setup for our recording prototype. We have mounted the recording system setup on a pickup truck and captured video outdoors in Moscow, Idaho, travelling at different speeds (max. 25 mph). During video capture, we frequently changed the camera view direction and zoom level. The captured video covered a 6km by 5km size region quite uniformly. However, for a few popular locations we shot several videos, each viewing the same location from different directions. The total captured data includes 140 videos, ranging from 60 to 240 seconds in duration. Figure 4 shows a visualization example of camera viewable scenes for two video files on a map. For visual clarity, viewable scene regions are drawn every three seconds. Due to space limitations we cannot include more example visualizations. Further samples can be found at *http://eiger.ddns.comp.nus.edu.sg/geospatialvideo/ex.html.*

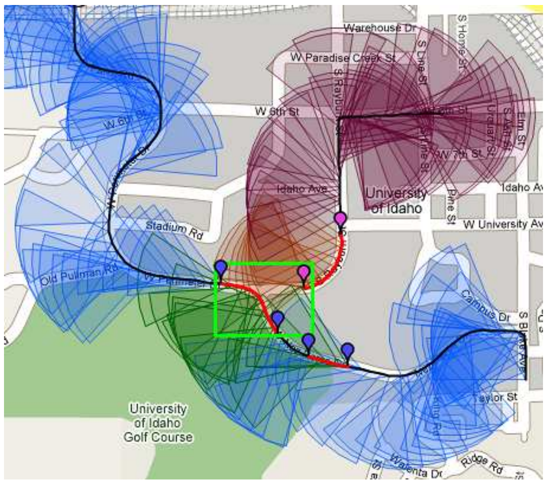


Figure 4: Visualization of viewable scenes on a map.

4.2 Analysis and Results

In this section we evaluate the performance of our scene based video search. We selected 250 random query regions (Q), of size 300m by 300m within the 6km by 5km area of total video coverage. We then searched the georeferenced video dataset for the videos that captured at least one of the query regions in Q and extracted the video segments that show these regions, i.e., where the viewable scene and the query region intersect. We specifically set out to answer the following questions:

- Does the proposed scene based video search algorithm return all matching videos? I.e., is the returned video result set *complete*?
- How well does the proposed video search algorithm effectively eliminate irrelevant video segments in a result video? I.e., how *accurate* is the search result?

The former argument is hard to verify, since there is no easy way to get the "ground truth" for the query result set. One possible way is to place easily recognizable objects in geo-space (e.g., a big green circular sign) to mark the boundaries of query regions. Then, using a feature extraction algorithm, the video streams can be processed to extract the video frames in which these objects are visible. However, such object recognition algorithms have their own limitations and suffer from low accuracy. Alternatively a human subject may watch all the clips and confirm whether a query region is visible in a video and mark the frames that show the region. Such manual verification is also prone to errors. However, human perception is still the most reliable source to determine whether an object is visible within a video. Although we could not perform the manual check for the whole dataset, we randomly chose 40 videos and had a student analyze them and mark the query regions that appear in any of these 40 videos. For the manual check, to eliminate human errors as much as possible, we clearly defined the conditions to conclude whether a query region is visible or not. First, to ease the process we adjusted the query boundaries to the closest road intersections, buildings or signs which are easy to recognize within the video. Second, we assured that, for the queries which are marked as visible, at least 10% of their region appears in some frame within the video. Also, we made sure that objects within query region (e.g., buildings) appear fairly clear within the video. If they are difficult to recognize, which means the query region is far away from the camera, it is not marked as visible. And third, if a query region is not visible due to an occlusion by a bigger object along the view direction, it is marked as invisible. Unfortunately, manual verification is quite laborious and requires careful scanning of videos. In our first step we removed the videos which are reasonably far away from each query region. In our experiments a query covers on average $1/400^{th}$, and each video $1/40^{th}$, of the total video coverage. Since query locations are uniformly distributed throughout the total covered area, such filtering pruned the number of result videos by 70% on average. As mentioned earlier, the query regions are associated with some landmarks within the geo-space and we were very familiar with this geographic region. For each query Q, we watched through the remaining videos and looked for those landmarks and the region delimited by them. If query region Q is clearly visible within the video, it is marked as visible by that video. Finally, we recorded the number of queries visible by each video.

To evaluate the accuracy of the query result set, we compared our approach with two other video scene description models as explained in Section 3.1: (1) The *CircleScene* model – the camera viewable scene is described as a circular region around the camera location with the assumption that the view direction is not known. A query is visible if its region intersects with the circular viewable scene. (2) The *PointScene* model - here the camera viewable scene is the camera location point. A query is visible if the camera point resides within the query's region.

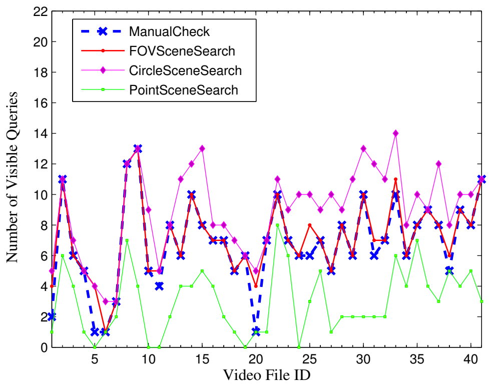


Figure 5: Number of visible queries per video file.

4.2.1 *Completeness of Result Video Set*

Given 250 random queries, through manual scan, we created the list of query regions that are visible within each video in the dataset of 40 videos. To verify the completeness, we needed to show that the proposed viewable scene based search algorithm does not miss any videos that actually intersect with Q. For that purpose, we executed our search algorithm on the same data set of 40 videos, using the same query set we used in the manual scan. We then repeated the search using *CircleSceneSearch* and *PointSceneSearch*. Figure 5 shows the number of queries marked as visible for each video file by (1) manual verification (*ManualCheck*) (2) the video search algorithm using the proposed *FOVScene* model (*FOVSceneSearch*) (3) the video search algorithm using the *CircleScene* model, ignoring view direction (*CircleSceneSearch*) and (4) the video search algorithm

	FOVSceneSearch vs *ManualCheck*	*CircleSceneSearch* vs *ManualCheck*	*PointSceneSearch* vs *ManualCheck*
MaxDiff	$\max_{\forall i, 1 \le i \le 40}(\lvert FOVR(i) - MR(i)\rvert) = 3.0\ (o)$	$\max_{\forall i, 1 \le i \le 40}(\lvert CR(i) - MR(i)\rvert) = 6.0\ (o)$	$\max_{\forall i, 1 \le i \le 40}(\lvert PR(i) - MR(i)\rvert) = 0.0\ (o)$
MinDiff	$\min_{\forall i, 1 \le i \le 40}(\lvert FOVR(i) - MR(i)\rvert) = 0.0\ (o)$	$\min_{\forall i, 1 \le i \le 40}(\lvert CR(i) - MR(i)\rvert) = 0.0\ (o)$	$\min_{\forall i, 1 \le i \le 40}(\lvert PR(i) - MR(i)\rvert) = 9.0\ (u)$
AvgDiff	$\frac{\sum_{\forall i, 1 \le i \le 40}(\lvert FOVR(i) - MR(i))\rvert}{\lvert MR(i)\rvert} = 0.27\ (o)$	$\frac{\sum_{\forall i, 1 \le i \le 40}(\lvert CR(i) - MR(i))\rvert}{\lvert MR(i)\rvert} = 2.10\ (o)$	$\frac{\sum_{\forall i, 1 \le i \le 40}(\lvert PR(i) - MR(i)\rvert)}{\lvert MR(i)\rvert} = 3.88\ (u)$
Precision	$\frac{\lvert FOVR(i) \bigcap MR(i)\rvert}{\lvert FOVR(i)\rvert} = 0.96$	$\frac{\lvert CR(i) \bigcap MR(i)\rvert}{\lvert CR(i)\rvert} = 0.77$	$\frac{\lvert PR(i) \bigcap MR(i)\rvert}{\lvert PR(i)\rvert} = 0.28$
Recall	$\frac{\lvert FOVR(i) \bigcap MR(i)\rvert}{\lvert MR(i)\rvert} = 1.0$	$\frac{\lvert CR(i) \bigcap MR(i)\rvert}{\lvert MR(i)\rvert} = 1.0$	$\frac{\lvert PR(i) \bigcap MR(i)\rvert}{\lvert MR(i)\rvert} = 0.12$

(o ←Overestimation, u ←Underestimation)

Table 2: Summary of metrics for evaluating accuracy (in terms of number of visible queries).

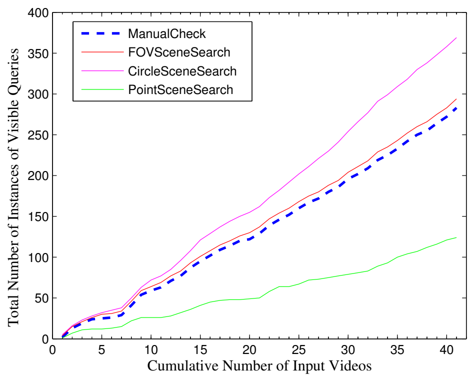


Figure 6: Cumulative number of visible queries as a function of number of input videos (40 videos only).

using the *PointScene* model (*PointSceneSearch*). Note that we used the same far visible distance (R) value for both the *CircleSceneSearch* and the *FOVSceneSearch* algorithms.

Results show that *FOVSceneSearch* correctly returns all videos marked visible through *ManualCheck*. However, the *FOVSceneSearch* result set also includes some false positives (i.e., returned as an intersecting query but the scene does not actually show the query region), which might occur due to the following reason: within the geo-space the camera view is sometimes occluded with structures, trees, cars, etc. Although the query region is within a fair distance from the camera and the camera points towards it, the query region can still be invisible in the video, therefore may not be included in the result set of the manual check. As expected, *CircleSceneSearch* returns an excessive percentage of extra irrelevant videos and overestimates the manual search results while *PointSceneSearch* underestimates the manual search results by returning only a subset of the visible queries. Let $MR(i)$, $FOVR(i)$, $CR(i)$ and $PR(i)$ be the sets of queries returned as visible for the i^{th} video file by algorithms *ManualCheck*, *FOVSceneSearch*, *CircleSceneSearch*, and *PointSceneSearch*, respectively. To quantify the amount of underestimation and overestimation of each search algorithm with respect to the manual search, we compare $FOVR(i)$, $CR(i)$ and $PR(i)$ to the manual result set $MR(i)$, $\forall i\ 1 \le i \le 40$. Table 2 shows the metrics for comparisons and summarizes the results. $MaxDiff$, $MinDiff$ and $AvgDiff$ are the maximum, minimum and average differences between the $FOVR$, CR and PR result sets and the MR result set, respectively. *Precision* and *Recall* are two metrics widely used in past studies for evaluating the degree of accuracy and comprehensiveness of a result set. Precision is the ratio of retrieved relevant items to all retrieved items. A lower value of precision implies the search result set contains a large number of invisible query regions. Recall is the ratio of retrieved relevant items to all relevant items. A lower recall value means more query regions that should be returned as visible are ignored.

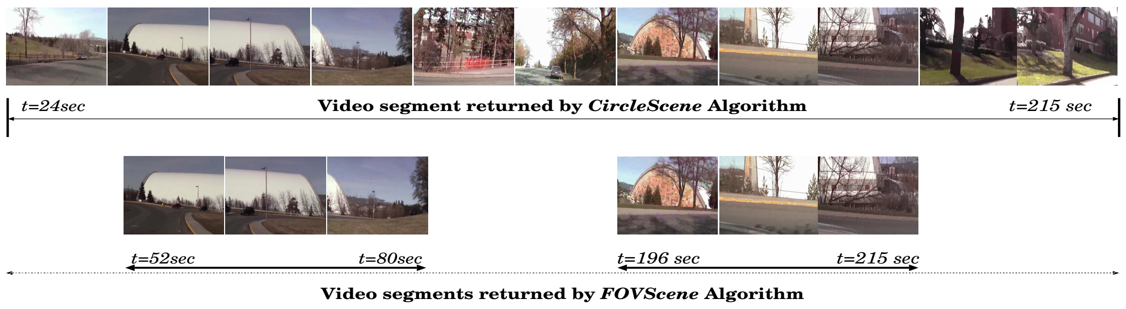
In Figure 5, we plot the exact number of intersecting queries for each approach, which results in a jaggy graph where the number of intersecting videos changes dramatically for each video file. To illustrate the overall performance of each algorithm better, Figure 6 shows the total number of instances[1] of intersecting queries identified and returned by each approach versus the total number of videos queried. Note that, in Figure 5, the horizontal axis shows the individual video files whereas in Figure 6, it shows the total number of video files included in the search. We observe that *CircleSceneSearch* returns a large percentage of false positives when compared to *ManualCheck*, whereas the proposed *FOVSceneSearch* closely matches the results of the manual check. The third approach, *PointSceneSearch* returns a very small percentage of the correct result set missing many intersecting videos.

We are aware that the completeness of the results from a single dataset may not imply the same for the general case. However, we argue that performing the search on a *fairly large* georeferenced video dataset acquired within a *fairly large real world geospace* without making specific assumptions and using a *large* and *random* query set serves as a realistic example which strongly resembles the generalized case.

4.2.2 Accuracy of Search Result

Both Figures 5 and 6 show the superiority of the *FOVScene* model over the *CircleScene* and *PointScene* models in georeferenced video search. The graphs illustrate the results using a subset (40 videos) of the gereferenced video data we collected. Figure 7 shows the cumulative sums of the query results for the whole dataset (140 videos). We used 250 random queries with a size of 300m by 300m square. The gap between search algorithms *FOVSceneSearch* and *CircleSceneSearch* increases as the input dataset becomes larger. It is important to note that, although a query region is marked as visible by both the *FOVScene* and the *CircleScene* mod-

[1]For example, when a query region intersects with three videos, we counted it as three instances of intersecting queries.



t=24sec
Video segment returned by *CircleScene* Algorithm
t=215 sec
t=52sec
t=80sec
t=196 sec
t=215 sec
Video segments returned by *FOVScene* Algorithm

Figure 10: Comparison of *CircleScene* and *FOVScene* results.

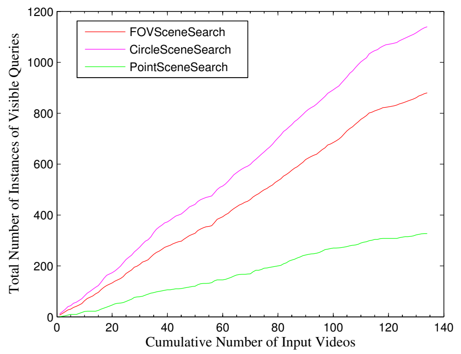


Figure 7: Cumulative number of visible queries as a function of number of input videos (whole dataset).

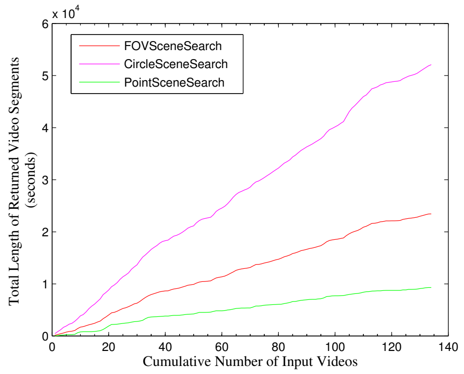


Figure 8: Cumulative sum of returned video segment lengths as a function of number of input videos (whole dataset).

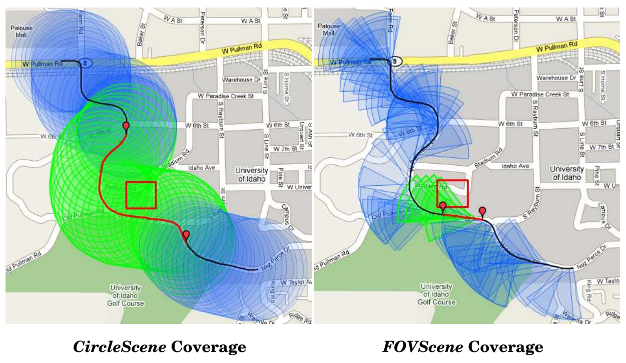


Figure 9: Comparison of *CircleScene* and *FOVScene* coverage. The square is the query region.

els, the video segments they report for the appearance of the query region in a single video can be different. For example, as shown in Figure 9, when the camera is rotated, although the query region is not visible anymore in the video, *CircleSceneSearch* will still report that the query intersects with its viewable scene for the following frames. Therefore, the total length of the video segments identified by *FOVSceneSearch* can be shorter (i.e., more precise with less false positives) than that by *CircleSceneSearch*. For comparison, in Figure 8, we plotted the total length of all video segments identified by each search algorithm while varying the number of input video files. Similar to Figure 7, we have used the cumulative sums to show the overall difference as the input data size grows.

When comparing Figures 7 and 8, we clearly see the difference in the gap between *FOVSceneSearch* and *CircleSceneSearch*. In Figure 7, the average percentage gain in accuracy for *FOVSceneSearch* over *CircleSceneSearch* is measured as 15.7% with a maximum difference of 21.4%. On the other hand, Figure 8 shows a 56.4% improvement in accuracy for the *FOVSceneSearch* result set. The maximum percentage difference in Figure 8 is 60.7%. These results show that the proposed *FOVSceneSearch* algorithm improves the accuracy of georeferenced video search not only by eliminating the irrelevant videos, but also by filtering out the video segments whose scenes do not contain the query region. For example, in Figure 10, *FOVSceneSearch* returns only subparts of the long video segment returned by *CircleSceneSearch*, eliminating the frames that do not show the query region. Considering the huge size of video data and time consuming human verification process for the final result, this significant reduction of false positives can greatly enhance the performance of video search. Therefore we conclude that, by using the camera view direction in addition to the camera location in viewable scene estimations, the accuracy of the search results can be improved dramatically while ensuring completeness of the result set.

To analyze the effect of the query size over the total length of the returned video segments, we repeated the same experiments shown in Figure 8 while varying the size of the query regions. Figure 11 reports the total length of the returned video segments by all three approaches for query region sizes ranging from 20m×20m to 550m×550m. For smaller query regions we observed bigger differences between the three approaches, i.e., the superiority of *FOVSceneSearch* is maximized for small query regions. The performance gap among the three approaches reduced as the query size increased. For sizes bigger than 550m×550m, we have not observed dramatic changes in the results.

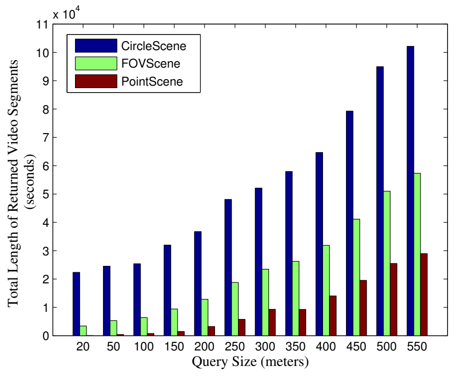


Figure 11: Effect of query size.

5. CONCLUSIONS AND FUTURE WORK

Our contributions in this paper have been threefold. First, we introduced a methodology for automatic annotation of video clips with a collection of meta-data such as camera location, viewing direction, field-of-view, etc. Such meta-data can provide a comprehensive model to describe the scene a camera captures. We proposed a *viewable scene* model that strikes a balance between the analytical complexity and the practical applicability of the scene description to enable effective and efficient search of videos. Second, we described our implemented prototype which demonstrates the feasibility of acquiring, storing, searching and retrieving metadata enhanced georeferenced video based on the proposed *viewable scene* model. We collected a sufficiently large set of georeferenced video data using our prototype system. Finally, we demonstrated the benefits of using our approach in accurately retrieving the relevant video segments for a given query. We plan to extend our work in several directions:

(i) In our initial work we used a simple relational database schema to store camera *viewable scenes*. We also mentioned some alternative spatio-temporal structures that can be used to index the area that a camera *viewable scene* covers. However, we argue that current work in spatio-temporal indexing can not fully optimize the search of a dynamically changing *viewable scene*. Therefore, there is a strong need for a better index structure that would specifically target georeferenced annotations of video data.

(ii) In our study we only show examples for simple spatial range queries. However, the proposed *viewable scene* model that includes the camera view direction and camera location provides a rich information base to answer more complex geospatial queries. Similarly, when query results are presented to a user, the resulting video segments can be ranked based on how relevant they are to the query requirements and user interests. In our initial work, we show how the search accuracy can be improved even for simple range queries using our *viewable scene* model. We will elaborate on video ranking in our future work.

(iii) There are several additional factors that influence the effective *viewable scene* in a video, such as occlusions, visibility depth, resolution, etc. The proposed *viewable scene* model has to be extended and improved to account for these factors. Occlusions have been well studied in computer graphics research. We plan to incorporate an existing occlusion determination algorithm into our model.

(iv) To enable video search on a larger scale, a standard format for georeferenced video annotations must be established and issues for enabling automated integration with other providers' data have to be investigated.

6. ACKNOWLEDGMENTS

This research has been funded in part by NSF grant IIS-0534761 and NUS AcRF grant WBS R-252-050-280-101/133. We also acknowledge the support of the NUS Interactive and Digital Media Institute (IDMI).

7. REFERENCES

[1] *Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.*

[2] *Flickr. http://www.flickr.com.*

[3] *Geobloggers. http://www.geobloggers.com.*

[4] *Woophy. http://www.woophy.com.*

[5] Boris Epshtein, Eyal Ofek, Yonatan Wexler, and Pusheng Zhang. Hierarchical Photo Organization Using Geo-Relevance. In 15th *ACM Intl. Symposium on Advances in Geographic Information Systems (GIS)*, pages 1–7, 2007.

[6] Clarence H. Graham, Neil R. Bartlett, John Lott Brown, Yun Hsia, Conrad C. Mueller, and Lorrin A. Riggs. *Vision and Visual Perception.* John Wiley & Sons, Inc., 1965.

[7] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD, Proceedings of Annual Meeting, Boston, Massachusetts*, pages 47–57, 1984.

[8] Eugene Hecht. *Optics.* Addison-Wesley Publishing Company, 4th edition, August 2001.

[9] Tae-Hyun Hwang, Kyoung-Ho Choi, In-Hak Joo, and Jong-Hun Lee. MPEG-7 Metadata for Video-Based GIS Applications. In *Geoscience and Remote Sensing Symposium*, pages 3641–3643, vol.6, 2003.

[10] Rieko Kadobayashi and Katsumi Tanaka. 3D Viewpoint-Based Photo Search and Information Browsing. In 28th *Intl. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 621–622, 2005.

[11] Kyong-Ho Kim, Sung-Soo Kim, Sung-Ho Lee, Jong-Hyun Park, and Jong-Hyun Lee. The Interactive Geographic Video. In *Geoscience and Remote Sensing Symposium*, pages 59–61, vol.1, 2003.

[12] Xiaotao Liu, Mark Corner, and Prashant Shenoy. SEVA: Sensor-Enhanced Video Annotation. In 13th *ACM Intl. Conference on Multimedia*, pages 618–627, 2005.

[13] Mor Naaman, Yee Jiun Song, Andreas Paepcke, and Hector Garcia-Molina. Automatic Organization for Digital Photographs with Geographic Coordinates. In 4th *ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 53–62, 2004.

[14] A. Pigeau and M. Gelgon. Building and Tracking Hierarchical Geographical & Temporal Partitions for Image Collection Management on Mobile Devices. In 13th *ACM Intl. Conference on Multimedia*, 2005.

[15] Kerry Rodden and Kenneth R. Wood. How do People Manage their Digital Photographs? In *SIGCHI Conference on Human Factors in Computing Systems*, pages 409–416, 2003.

[16] Rainer Simon and Peter Fröhlich. A Mobile Application Framework for the Geospatial Web. In 16th *Intl. Conference on the World Wide Web*, pages 381–390, 2007.

[17] Yannis Theodoridis, Michael Vazirgiannis, and Timos Sellis. Spatio-Temporal Indexing for Large Multimedia Applications. In *IEEE Intl. Conf. on Multimedia Systems*, 1996.

[18] Carlo Torniai, Steve Battle, and Steve Cayzer. *Sharing, Discovering and Browsing Geotagged Pictures on the Web.* Springer, 2006.

[19] Kentaro Toyama, Ron Logan, and Asta Roseway. Geographic Location Tags on Digital Images. In 11th *ACM Intl. Conference on Multimedia*, pages 156–166, 2003.