Vector Model in Support of Versatile Georeferenced Video Search

Seon Ho Kim
University of the District of Columbia
Washington, DC 20008
skim@udc.edu

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

Byunggu Yu
University of the District of Columbia
Washington, DC 20008
byu@udc.edu

Roger Zimmermann
National University of Singapore
Singapore 117417
rogerz@comp.nus.edu.sg

ABSTRACT

Increasingly geographic properties are being associated with videos, especially those captured from mobile cameras. The meta data from camera-attached sensors can be used to model the coverage area of the scene as a spatial object such that videos can be organized, indexed and searched based on their field of views (FOV). The most accurate representation of an FOV is through the geometric shape of a circular sector. However, spatial search and indexing methods are traditionally optimized for rectilinear shapes because of their simplicity. Established methods often use an approximation shape, such as a minimum bounding rectangle (MBR), to efficiently filter a large archive for possibly matching candidates. A second, refinement step is then applied to perform the time-consuming, precise matching function. MBR estimation has been successful for general spatial overlap queries, however it provides limited flexibility for georeferenced video search. In this study we propose a novel vector-based model for FOV estimation which provides a more versatile basis for georeferenced video search while providing competitive performance for the filter step. We demonstrate how the vector model can provide a unified method to perform traditional overlap queries while also enabling searches that, for example, concentrate on the vicinity of the camera's position or harness its view direction. To the best of our knowledge no comparable technique exists today.

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

ACM MMsys'10, February 22–23, 2010, Phoenix, Arizona, USA.

1. INTRODUCTION

Advances in sensor technologies allow video clips to be tagged with geographic properties, such as camera locations from GPS and camera directions from digital compasses, while being collected. Importantly, such meta-data can be attached to the video streams automatically, hence allowing for the consistent annotation of large amounts of collected video contents and thus enabling various criteria for versatile video search. The captured geographic meta-data have a significant potential to aid in the indexing and searching of georeferenced video data at the high semantic level preferred by humans. However, there has been little research on utilizing such meta-data for the systematic indexing and searching of video data.

Some video data are naturally tied to geographic locations. For example, video streams from traffic monitoring may not have much meaning without their associated location information. Thus, associated applications typically let a user specify location information to retrieve the traffic video related to a point or region. In anticipation of future applications, more and more still images are automatically tagged with geographic data as high-end devices are equipped with various sensors. Example cameras include the Sony GPS-CS1, the Nikon D90 with GPS; the Ricoh SE-3 and the Solmeta DP with GPS plus compass. Many smartphones are now equipped with a camera, GPS and accelerometer. Recent camcorders with a built-in GPS receiver (e.g., Sony HDR-XR520V) allow automatic geo-tagging for both still images and videos. The resulting fused video plus sensor data streams can provide an effective means to index and search videos, especially for large archives that handle an extensive amount of video data.

One class of video search techniques has naturally focused on identifying objects within the captured content through sophisticated extraction methods at the signal level or mining associated textual meta-data description. Even now, the geo-tagging data are mostly used for organizing or grouping images based on location information in a simple and straightforward way. Furthermore, most videos captured are not panoramic and as a result the viewing direction becomes very important for human perception, and consequently for video searching. GPS data only identifies object locations and therefore it is imperative to investigate the natural concepts of a human viewing direction and a view point. For example, we may be interested in videos that show a building only from a specific angle. The question arises whether a video database search can accommodate such human friendly concepts, i.e., whether it is possible

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.

The collection and fusion of multiple sensor streams such as the camera location, the field-of-view (FOV), the direction, etc., can provide a comprehensive model of the viewable scene. For example, one may model the viewable scene area (i.e., FOV) of video segments or frames using a pie-shaped geometric contour, i.e., a pure spatial object, thus transforming the video search problem into a spatial data selection problem. The objective then is to index the spatial objects and to search videos based on the geographic properties of videos. Beyond the importance of the geographic information where a video is taken, there are other obvious advantages in exploiting the spatial properties of video because the operation of a camera is fundamentally related to geometry. When a user wants to find images of an object captured from a certain viewpoint and from a certain distance, these semantics can be interpreted as geometric relations between the camera and the object such as the Euclidean distance between them and the directional vector from the camera to the object. Thus, more meaningful and recognizable results may be achieved by using spatial queries on georeferenced videos.

In this study we propose a new vector-based approximation model for efficient indexing and searching of georeferenced video based on an FOV model. The FOV model represents a viewable scene of images as a circular sector using the tagged camera location, the direction, the angle, and the maximum viewable distance as illustrated in Figure 2. As we will show, this model provides a more versatile basis for georeferenced video search while providing competitive performance. We demonstrate how the vector model can provide a unified method to perform traditional overlap queries while also enabling searches that, for example, concentrate on the vicinity of the camera's position or take into account its view direction. To the best of our knowledge no comparable technique exists today.

2. RELATED WORK

Associating GPS coordinates with digital media (images and videos) has become an active area of research [15]. There has been significant research on organizing and browsing personal photos according to location and time. Toyama et al. [18] 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. A number of additional techniques in this direction have been proposed [12, 14]. There are also several commercial web sites [1, 2] that allow the upload and navigation of georeferenced photos. All these techniques use only the camera geo-coordinates as the reference location in describing images. We instead rely on the field-of-view of the camera to describe the scene. More related to our work, Ephstein et al. [6] proposed to relate images with their view frustum (viewable scene) and used a scene-centric ranking to generate a hierarchical organization of images. Several additional methods are proposed for organizing [16, 9] and browsing [7, 17] images based on camera location, direction and additional meta-data. Although these research work are similar to ours in using the camera field-of-view to describe the viewable scene, their main contribution is on image browsing and grouping of similar images together. Some approaches [17, 10] use location and other meta-data, as well as tags associated with images, and the images' visual features to generate representative candidates within image clusters. Geo-location is often used as a filtering step. Some techniques [6, 16] solely use camera location and orientation in retrieving the "typical views" of important objects. However then the emphasis is on the segmentation of image scenes and organizing photos based on image scene similarity. 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.

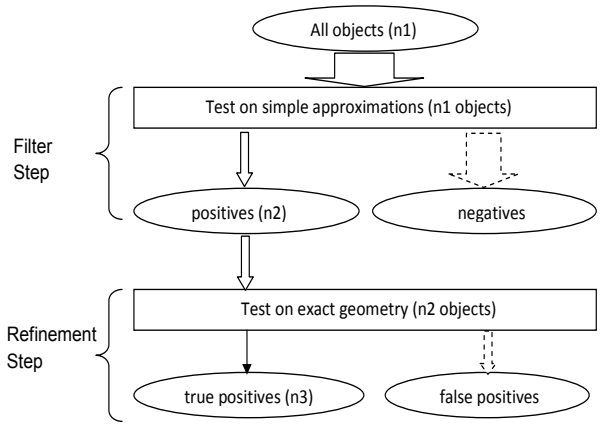


Figure 1: Illustration of filter-refinement steps.

There exist only a few systems that associate videos with their corresponding geo-location. Liu et al. [11] 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 geo-spatially annotate videos for effective video search. Our prior work [3] investigated these issues and proposed a viewable scene model to describe the video content but did not address the search issues.

3. MODELING FOV USING VECTOR

3.1 Motivation

When a large collection of videos is stored in a database, the cost of processing spatial queries may be significant because of the computational complexity of the operations involved. Therefore, such queries are typically executed in two steps: a *filter* step followed by a *refinement* step [13, 5] (Figure 1). The idea behind the filter step is to approximate the large number of complex spatial shapes (n1 objects in Figure 1) with simpler outlines (e.g., a minimum bounding rectangle, MBR [4]) so that a large number of unrelated objects can be dismissed very quickly based on their simplified shapes. The resulting candidate set (n2 objects) is then further processed during the refinement step to determine the exact results (n3 objects) based on the exact geometric shapes. The rationale of the two step process is that the filter step is computationally far cheaper than the refinement step due to the simple approximations. Overall, the cost of spatial queries is determined by the efficiency of the filter

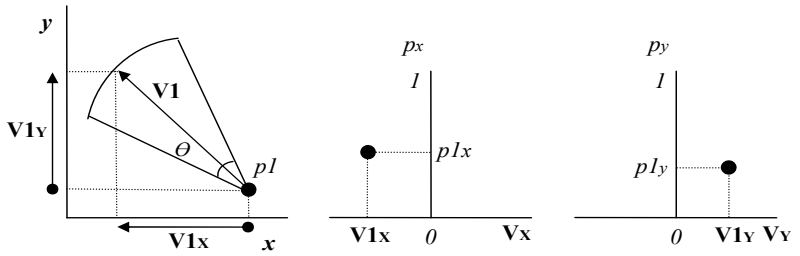


Figure 2: FOV representation in different spaces.

step (many objects, but simple shapes) and the complexity of the refinement step (few objects with complex shapes).

Additionally, in video search applications, the refinement step can be very expensive due to the nature of the processing. Depending on the application, various computer vision and content-based extraction techniques may be applied before presenting the search results. For example, some occlusions may need to be detected based on local geographic information such as the location and size of buildings. Some specific shapes or colors of objects might be analyzed for more accurate results, or the quality of images such as brightness and focus may be considered in determining the relevance ranking of results. Such extra processing is in general performed during refinement on a per frame basis, therefore significantly increases the time and execution cost of the refinement step. It is thus critical to minimize the amount of refinement processing for large scale video searches. This, in turn, motivates the use of effective and efficient filtering algorithms which minimize the number of frames that need to be considered in the refinement step.

In traditional spatial data processing, MBR approximations are very effective for the filter step. However, with a bounding rectangle some key properties that are useful in video search applications may be lost. For example, MBRs retain no notion of directionality. This study advocates a new vector approximation that provides similar efficiency and low processing cost as MBR-based methods, but additionally provides better support for the type of searches that a video database may encounter. Thus, the main focus of the paper is to provide a novel filter step called the *vector model* as a more efficient and effective filter step for the large scale georeferenced video search applications and to provide its comparison to a conventional filter step using MBRs. An identical refinement step will be assumed for a fair comparison between the vector and MBR model.

In the following sections we will introduce our vector model and illustrate that it is both competitive with MBR-based methods where applicable, but also extends to cases that MBRs cannot handle.

3.2 Vector Model

A camera positioned at a given point p in geo-space captures a scene whose covered area is referred to as camera field-of-view (FOV, also called a viewable scene). The metadata related to the geographic properties of a camera and its captured scenes are as follows: 1) the camera position p consists of the latitude, longitude coordinates read from a positioning device (e.g., GPS), 2) the camera direction α is obtained based on the orientation angle ($0^\circ \leq \alpha < 360^\circ$) provided by a digital compass, 3) the maximum visible distance from p is R at which objects in the image can be recognized by observers [3] – since no camera can capture meaningful images at an indefinite distance, R is bounded by M which is the maximum distance set by an application –, and 4) the camera view 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 [8]. The above geo-properties are captured from a sensor-equipped camera while video is recorded.

Based on the availability of the sensor input data, the FOV of a video frame forms an area of circular sector shape (or pie-slice shape) in 2D geo-space as shown in Figure 2. Then, an FOV can be represented as a tuple $<T, p, \theta, \mathbf{V}>$, with T as the real time when the frame was captured, a position p, an angle θ, and a center vector $\mathbf{V}$. The magnitude of $\mathbf{V}$ is the viewable distance from p, i.e., R and the direction of $\mathbf{V}$ is α.

For indexing purposes, we propose a vector estimation model that represents an FOV using only the camera position p and the center vector $\mathbf{V}$. When we project the FOV onto the x and y axis, a point p is divided into p_x and p_y, and $\mathbf{V}$ is divided into V_X and V_Y along the x and y axis, respectively. Then, an FOV denoted by a point and vector can be represented by a quadruple $<p_x, p_y, V_X, V_Y>$; this can be interpreted as a point in four dimensional space.

In mathematics, space transformation is an approach to simplify the study of multidimensional problems by reducing them to lower dimensions or by converting them into some other multidimensional space. Using a space transformation, an FOV $<p_x, p_y, V_X, V_Y>$ can be divided and represented in two 2D subspaces, i.e., $p_x - V_X$ and $p_y - V_Y$. Then, an FOV can be represented as two points, each in its own 2D space. For example, Figure 2 shows the mapping between an FOV represented by $p1$ and $\mathbf{V1}$ in geo-space and two points in two transformed spaces without loss of information. To define the vector direction, let any vector heading towards the right (East in the northern hemisphere) on the x axis have a positive V_X value, and a negative V_X value for the other direction (West). Similarly, any vector heading up (North) on the y axis has a positive V_Y value, and a negative V_Y value for the other direction (South). Using the proposed model, any single FOV can be represented as a point in a $p - V$ space. As a result, the problem of searching for FOV areas in the original space can be converted to the problem of finding FOV points in the transformed subspace.

Note that the actual FOV is an area represented by a circular sector, so representing an area using a single vector is incomplete. More precisely, the FOV can be considered as a collection of vectors starting from p to all the points on the arc. To simplify the discussion for now we use only one

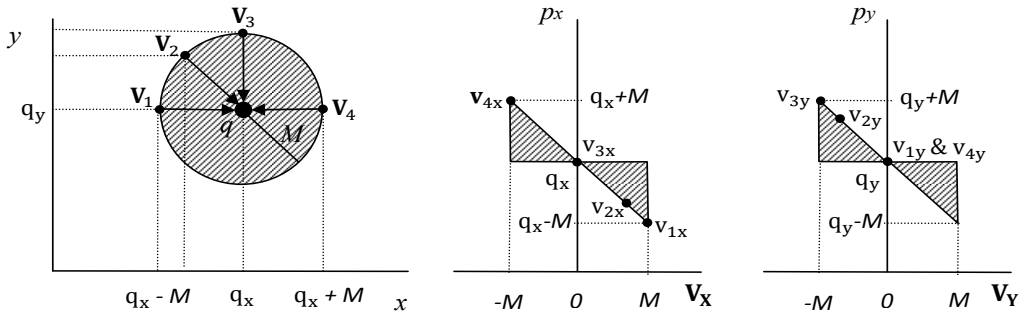


Figure 3: Illustration of filter step in point query processing.

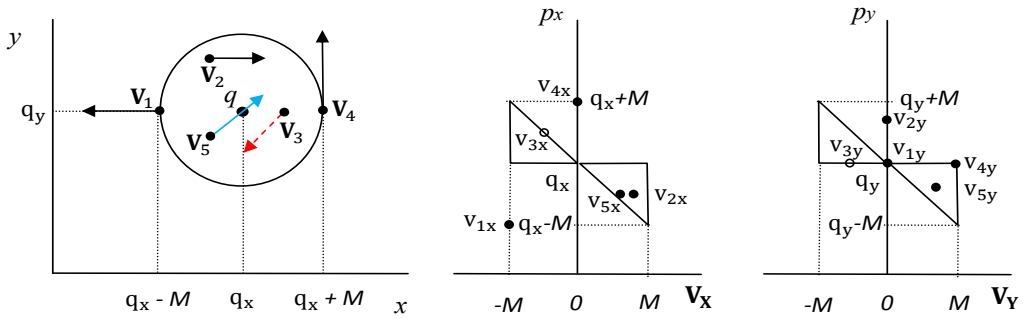


Figure 4: Example of filtering in point query.

center vector to represent an area as described above. We will relax this simplifying assumption in Section 5.

4. QUERY PROCESSING

When we represent video content as a series of FOVs which have a specific shape, FOVs can be considered as spatial objects. The problem of video search is then transformed into finding spatial objects in a database. This section describes how the filter step can be performed by using the proposed vector model for some typical spatial query types.

4.1 Point Query

The assumed query is, "For a given query point $q < x, y >$ in 2D geo-space, find all video frames that overlap with q." The filter step can be performed in $p-V$ space by identifying all possible points of FOVs that have a potential to overlap with the query point.

Recall that the maximum magnitude of any vector is limited to M, and hence any vector outside of a circle centered at the query point q with a radius M cannot reach q in geo-space; see Figure 3 for an illustration. Only vectors starting inside the circle (including the circumference of the circle) have the possibility to cross or meet q. Because a query point is not a vector, it is mapped only to the p axis. First, let us consider only the x components of all vectors. In $p_x - V_X$ space, the possible vectors that can cross (or touch) q_x should be in the range $[q_x - M, q_x + M]$. That is, any vector at p_x is first filtered out if $|p_x - q_x| > M$. Next, even though a vector is within the circle, it cannot reach q_x if its magnitude is too small. Thus, $|p_x - q_x| \leq |V_X|$ must be satisfied for V_X to reach q_x. At the same time the vector direction should be towards q_x. For example, when $p_x > q_x$, any vector with a positive V_X value cannot meet q_x. Hence, in $p - V$ spaces as shown in Figure 3, all points (i.e., all vectors) outside of the shaded isosceles right triangle areas will be excluded in the filter step. For example, vector V_1 in geo-space is represented as a point v_1 in $p - V$ space. Now consider all vectors starting from a point on the circumference of the circle towards the center with the maximum magnitude M. All such vectors moving from V_1 to V_4 in a clockwise direction map to the diagonal line starting from v_1 to v_4 in $p - V$ space. The same can be observed for the y components of vectors, i.e., the same shape appears in $p_y - V_Y$ space. The resulting vectors from the filter step should be included in the shaded areas of both $p_x - V_X$ and $p_y - V_Y$ space. Formally, a vector at p that satisfies the following conditions can be selected in the filter step:

$$\begin{array}{rll} |p - q| \leq M & & \\ p_x - q_x \leq -V_X & \text{if} & p_x > q_x \\ p_y - q_y \leq -V_Y & \text{if} & p_y > q_y \\ q_x - p_x \leq V_X & \text{if} & q_x > p_x \\ q_x - p_y \leq V_Y & \text{if} & q_y > p_y \\ any\ V_X & \text{if} & q_x = p_x \\ any\ V_Y & \text{if} & q_y = p_y \end{array} \quad (1)$$

Figure 4 shows five examples of FOVs and their mapping between $x - y$ space and $p - V$ spaces. The starting points of all five vectors are within the circle. However, not all of them pass the filter step. The starting points of three vectors, V_1, V_2, and V_4, are located inside the circle but their vector direction and/or magnitude do not meet the necessary conditions so they are filtered out. For example, V_{1x} is heading in the opposite direction even though its magnitude is large enough. Thus, v_{1x} is outside of the triangle shape search space. Similarly, V_{4x} is heading in the wrong direction so v_{4x} is outside of the search space. V_5 is directly heading towards q and both V_{5x} and V_{5y} have a large enough magnitude to reach q. Thus, both v_{5x} and v_{5y} are inside the

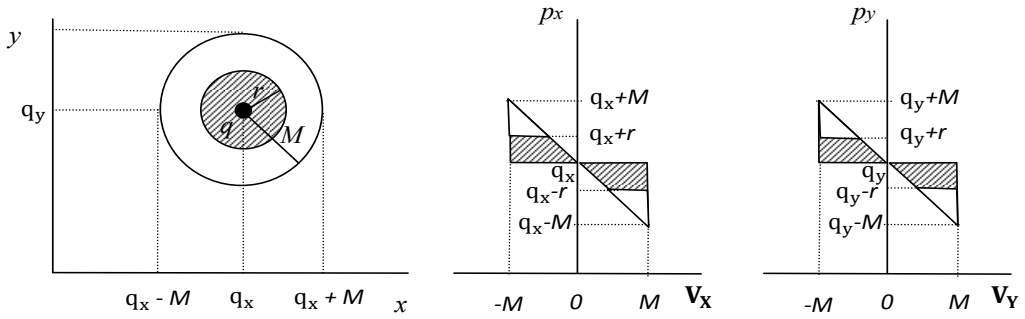


Figure 5: Illustration of the filter step in point query with bounded distance r.

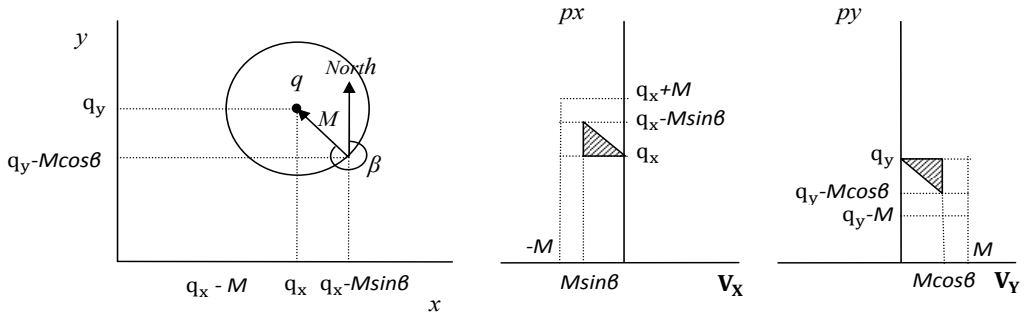


Figure 6: Illustration of filter step in directional point query with angle β.

search space, which means the vector should be included in the filter result. V_3 is considered a false positive in the filter result because it satisfies the conditions but actually does not cover q. It will be pruned out in the refinement step.

4.2 Point Query with Bounded Distance

Unlike with a general spatial query, video search may enforce application specific search parameters. For example, one might want to retrieve only frames where a certain small object at a specific location appears within a video scene, but with a given minimum size for better visual perception. Usually, when the camera is close to the query object, the object appears larger in the frame. Thus, we can devise a search with a range restriction for the distance of the camera locations from the query point such as "For a given query point $q < x, y >$ in 2D geo-space, find all video frames that overlap with q and that were taken within the distance r from q." Because of the distance requirement r, the position of the camera in an FOV cannot be located outside of the circle centered at q with radius r, where $r < M$. Thus, the search space can be reduced as shown in Figure 5.

4.3 Directional Point Query

The camera view direction can be an important factor for the image perception by an observer. Consider the case where a video search application would like to exploit the collected camera directions for querying. An example search is, "For a given query point $q < x, y >$ in geo-space, find all video frames taken with the camera pointing in the Northwest direction and overlapping with q." The view direction can be defined as a line of sight from the camera to the query point (i.e., an object or place pictured in the frame). The line of sight can be defined using an angle at the camera location similar to the camera direction α. Note that the camera orientation is always pointing to the center of an FOV scene while the view direction can point to any locations or objects within the scene. A digital compass mounted on a camera will report the camera direction primarily using bearings. A bearing is a horizontal angle measured clockwise from North (either magnetic North or true North) to a specific direction. When we use bearing as the view direction angle (say β), the Northwest direction is equivalent to 315 degrees (Figure 6). An important observation is that all FOVs that cover the query point have their starting points along the same line of sight in order to point towards the requested direction. Thus, the filter step needs to narrow the search to the vectors that satisfy the following conditions: 1) their starting points are on the line of sight, 2) their vector directions are heading towards q, and 3) their vector magnitudes are long enough to reach q.

For a given view direction angle β, we can calculate the maximum possible displacement of a vector starting point from the query point. Because the largest magnitude of any vector is M, the maximum displacement between the query point and the starting point of any possible overlapping vector is $-M \sin\beta$ on the x axis and $-M \cos\beta$ on the y axis (note that the sign is naturally decided by β, e.g., $\sin 315^\circ = -0.71$ and $\cos 315^\circ = 0.71$). In other words, as shown in Figure 6, any vector starting at a point greater than $q_x + (-M \sin\beta)$ on the x axis or less than $q_y + (-M \cos\beta)$ on the y axis cannot touch or cross the query point with the given angle β. Thus, the search area for such vectors can be reduced as illustrated in Figure 6. To meet the view direction request (say, 315° line of sight), no vector with a positive V_X value can reach q. Therefore, in the filter step the entire search space (i.e., the triangle shape) on the positive V_X side is excluded in the $p_x - V_X$ space. Similarly, no vector with a negative V_Y value can reach q, so the en-

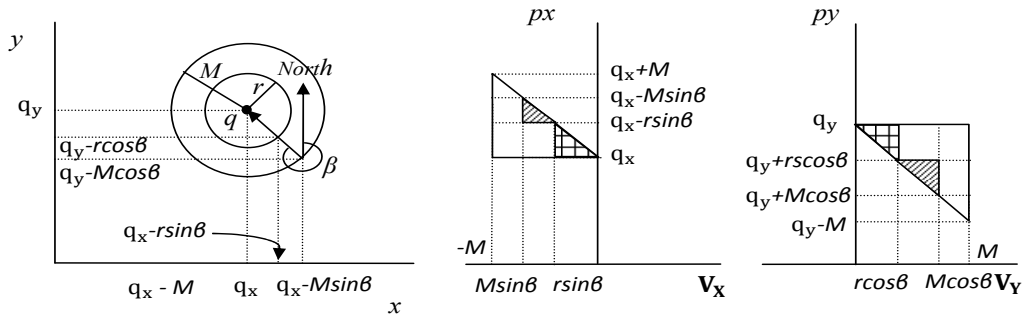


Figure 7: Illustration of filter step in directional point query with β and r.

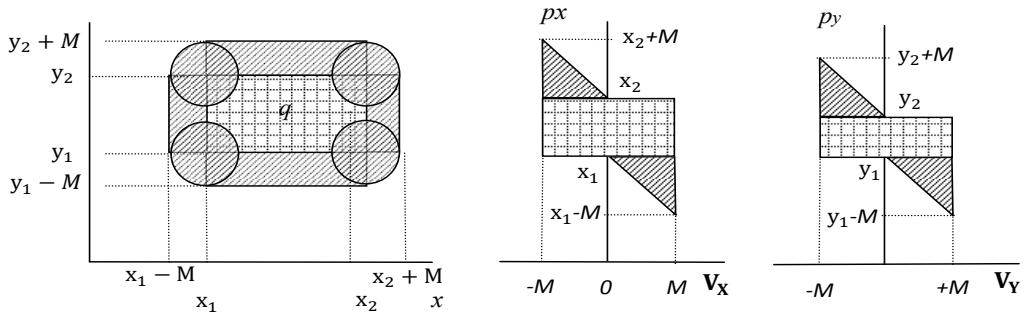


Figure 8: Illustration of filter step in range query.

tire search space (the triangle shape) on the negative V_Y side is excluded in the $p_x - V_Y$ space. Next, the size of the remaining search space is reduced because the range of possible V_X and V_Y values is now $[0, M \sin\beta]$ and $[0, M \cos\beta]$, respectively.

Using only a single specific view direction value may not be practical in video search because a slight variation in view directions does not significantly alter the human visual perception. Therefore, it will be more meaningful when the query is given with a certain range of directions such as $\beta \pm \epsilon$, e.g, $315^\circ \pm 10^\circ$. The extension can be straightforward and it will increase the search area in the $p - V$ space.

4.4 Directional Point Query with Bounded Distance

This type of query is a hybrid of the previous types. For a very specific search, the user might specify the query position, the view direction from the camera, and the distance between the location of the query and the camera. An example query is, "For a given query point $q < x, y >$ in geo-space, find all video frames heading in the Northwest direction, overlapping with q and taken within the distance r from q." The objective of this query is to find frames in which small objects (e.g., a 6 meter-high statue) at the query point appear large in the viewable scenes. Another example query is, "For a given query point $q < x, y >$ in geo-space, find all video frames heading in the Northwest direction that overlap with q and that were taken farther than the distance r from q." Now the intention is to find frames where large objects (e.g., a 6 story-tall building) at the query point appear prominently in the frames. For the former case, the positions of cameras are bounded by $-r \sin\beta$ from q_x on the p_x axis and $-r \cos\beta$ from q_y on the p_y axis, respectively. At the same time, the vector is bounded by $r \sin\beta$ on the V_X axis and $r \cos\beta$ on the V_Y axis, respectively. Therefore, the grid patterned triangle area in Figure 7 represents the search space. For the latter case, the positions of cameras are bounded within $[-r \sin\beta, -M \sin\beta]$ on the p_x axis and $[-r \cos\beta, -M \cos\beta]$ on the p_y axis, respectively. Furthermore, the vector is bounded within $[r \sin\beta, M \sin\beta]$ on the V_X axis and $[r \cos\beta, M \cos\beta]$ on the V_Y axis, respectively. Therefore, the shaded triangle area in Figure 7 represents the search space.

4.5 Rectangular Range Query

The assumed query is, "For a given rectangular query range in geo-space, find all the video frames that overlap with this region." Assume that the rectangular query region q is a collection of points (the rectangular shaded area with a grid pattern in Figure 8). When we apply the same space transformation, all points in the query region can be represented as a line interval on the p_x and p_y axes. First, when any vector's starting point falls inside the query region, the vector clearly overlaps with q so it should be included in the result of the filter step. Next, when we assume that any location along the perimeter of q is an independent query point as in Section 4.1, the starting points of vectors that can reach the query point is bounded by a circle with radius M. Drawing circles along all the points on the perimeter forms the shaded region in Figure 8. It follows that any vector with its starting point outside of the shaded region cannot reach any point in q. Only vectors starting inside the region have a possibility to cross q.

The search area in the $p - V$ spaces can be defined as shown in Figure 8. Again, any vector in the resulting set should be found in both search areas in the $p_x - V_X$ and $p_y - V_Y$ spaces. When p_x and p_y of a vector fall inside the mid-rectangles (grid pattern), p is inside q so the vector

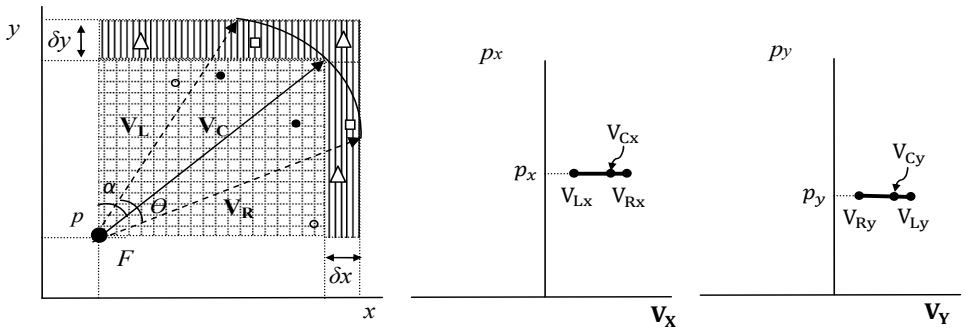


Figure 9: Problem of single vector model in point query processing.

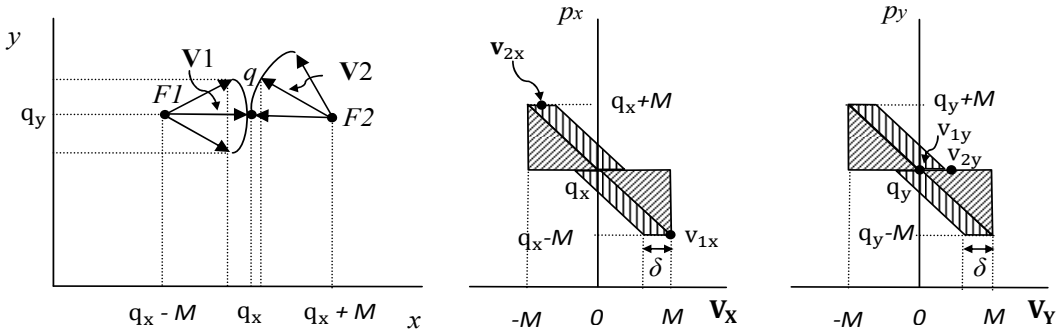


Figure 10: Overestimation constant δ.

automatically overlaps with q regardless of its direction or magnitude. However, when p is located outside of q, the vector's direction and magnitude should be considered to determine the overlap.

5. IMPLEMENTATION

So far, we assumed that an FOV is represented by a single center vector. However, in reality, an FOV is a collection of vectors with the following properties: 1) they all start from the same point, 2) they have the same magnitude |V|, and 3) they have different directions to points along the arc of a circular sector. In this paper we define an FOV using $<T, p, \theta, V>$, where V is the center vector of a FOV (i.e., V_C in Figure 9). V_C consists of a compass bearing α as direction and the visible distance R as magnitude. When only a single vector V_C represents the entire area of an FOV, there is a limitation in retrieving all the objects covered by the FOV. Because V_{CX} and V_{CY} are used to represent the FOV in $p - V$ spaces as described in Section 3, this approach underestimates the coverage of the FOV. In Figure 9, the rectangle with the grid pattern represents the estimation of the FOV in the filter step using V_C. Only query points inside the rectangle are selected during the filter step. The black dots overlap with the actual FOV so they represent the true query results. The white dots overlap with the estimation of the FOV but they are not actually overlapping with the FOV. The single vector model cannot exclude these points during the filter step, thus they become false positives. The problem is that the white rectangles are filtered out even though they are actually inside the FOV. They are completely missed during the search.

Alternatively, one can use two vectors to represent an FOV, the leftmost and the rightmost vector (V_L and V_R). Both have the same magnitude but different directions (their calculation from the collected data V_C is straightforward). When we use V_L and V_R to estimate the FOV, the estimation area is extended by δ_x and δ_y along the x and y axis, respectively (grid rectangle plus shaded areas in Figure 9). This approach can encompass the black dots, the white dots, and the white rectangles, which means that it is not missing any query points within the FOV. However, the number of false positives may also increase due to the bigger estimation area. The triangular points in the figure become false positives which are filtered out in the single vector model. Note that the two-vector model now has an identical estimation size compared with the MBR model. The more serious problem is that the two-vector model makes the use of $p-V$ space as search region more complex because we cannot use a simple point query in $p-V$ space. This is because an FOV is represented by a line interval bounded by the two vectors, as seen in Figure 9. The exact boundary is related to the camera direction α, the angle θ, and the visible distance R.

This problem can be resolved when we introduce an overestimation constant δ in defining the search area in $p - V$ space. The overestimation constant is a generalization of errors in using a single vector model, i.e., δ_x and δ_y. As shown in Figure 10, a single vector **V1** represents an FOV, F1. This vector covers the query point in the middle of an FOV and so it can be searched without any problem using the triangular shaped search space as originally described in Section 4.1. However, the other vector **V2**, representing F2, cannot be included in the search space. Because the query point is located at the leftmost corner of F2, v_{2y} covers q_y but v_{2x} falls outside of the triangle. **V2** is not considered as overlapping so F2 is missed. However, if the search space is

extended by δ along the V axis (the parallelogram-shaped shaded area), v_{2x} becomes included in the search space and **V2** can be selected in the filter step. Note that, in Figure 10, δ is applied in one direction because the other direction already reaches the maximal value of M. The next question is how to define the overestimation constant δ.

The overestimation constant can be determined by the tolerable error between the magnitude of the center vector and the leftmost (or rightmost) vector as explained above. Assuming a regular camera (non-panoramic), the camera angle θ can be 180° in the worst case, which results in the maximum difference M. This maximum value, i.e., $\delta = M$, significantly increases the search area in $p - V$ space. As θ becomes smaller, the extended search area decreases. The range of the overestimation constant is $0 \leq \delta \leq M$. However, note that normal camera lenses generally cover between 25° to 60° and wide angle lenses cover between 60° to 100°. Only ultra wide angle lenses capture up to 180°.

An interesting observation on the overestimation constant is that it can be an important parameter of georeferenced video search. First, the angle θ is related to the zoom level of the camera and the visible distance R. For a certain angle θ, the overestimation constant is limited to $M \sin(\theta/2)$ for 100% coverage. In our experiments, the widest measured angle was 60° and the maximum visible distance was 259 meters. In this case the worst overestimation constant will be $259 \times \sin(60/2) = 129.5$ meters. Another important observation in video search is that small objects which cannot be easily perceived by humans may be sometimes ignored even though they actually appear in FOVs. For example, if an object appears in the far left corner of an FOV and occupies only a very small portion of the frame, users may not be interested in such results. Moreover, if an object is located very far from the camera location (i.e., near the arc in our proposed model), it might be blocked by some nearer objects. Different applications (or users) might require different levels of accuracy in search results. So the overestimation constant provides a tradeoff between the performance and the accuracy of video search. Note that a smaller overestimation constant finds FOVs where the query point appears in the center part of the frames and effectively discriminates against other frames where the query point appears towards the far left or far right side of the frames.

6. EXPERIMENTAL EVALUATION

6.1 Data Collection and Methodology

To collect real georeferenced video data, we have constructed a prototype system which includes a camera, a 3D compass and a GPS receiver. We used the JVC JY-HD10U camera with a frame size of approximately one megapixel (1280×720 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 precise tilt compensated headings with roll and pitch data. To acquire the camera location, the Pharos iGPS-500 GPS receiver has been used. A program was developed to acquire, process, and record the georeferences along with the MPEG-2 HD video streams. The system can process MPEG-2 video in real-time (without decoding the stream) and each video frame is associated with its viewable scene information. In all of our experiments, an FOV was constructed every second, i.e., one FOV per 30 frames of video.

We mounted the recording system setup on a vehicle and captured video driving along streets at different speeds (max. 25 MPH). During video capture, we frequently changed the camera view direction. The recorded videos covered a 5.5 kilometer by 4.6 kilometer 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 134 video clips, ranging from 60 to 240 seconds in duration. Each second, an FOV was collected, resulting in 10,652 FOVs in total. We generated 1,000 point queries which were randomly distributed within the region. Figure 11.(a) shows the distribution of the camera positions of 10,652 FOVs and the 1,000 query points. For each query, we searched the georeferenced meta-data to find the FOVs that overlap with that query.

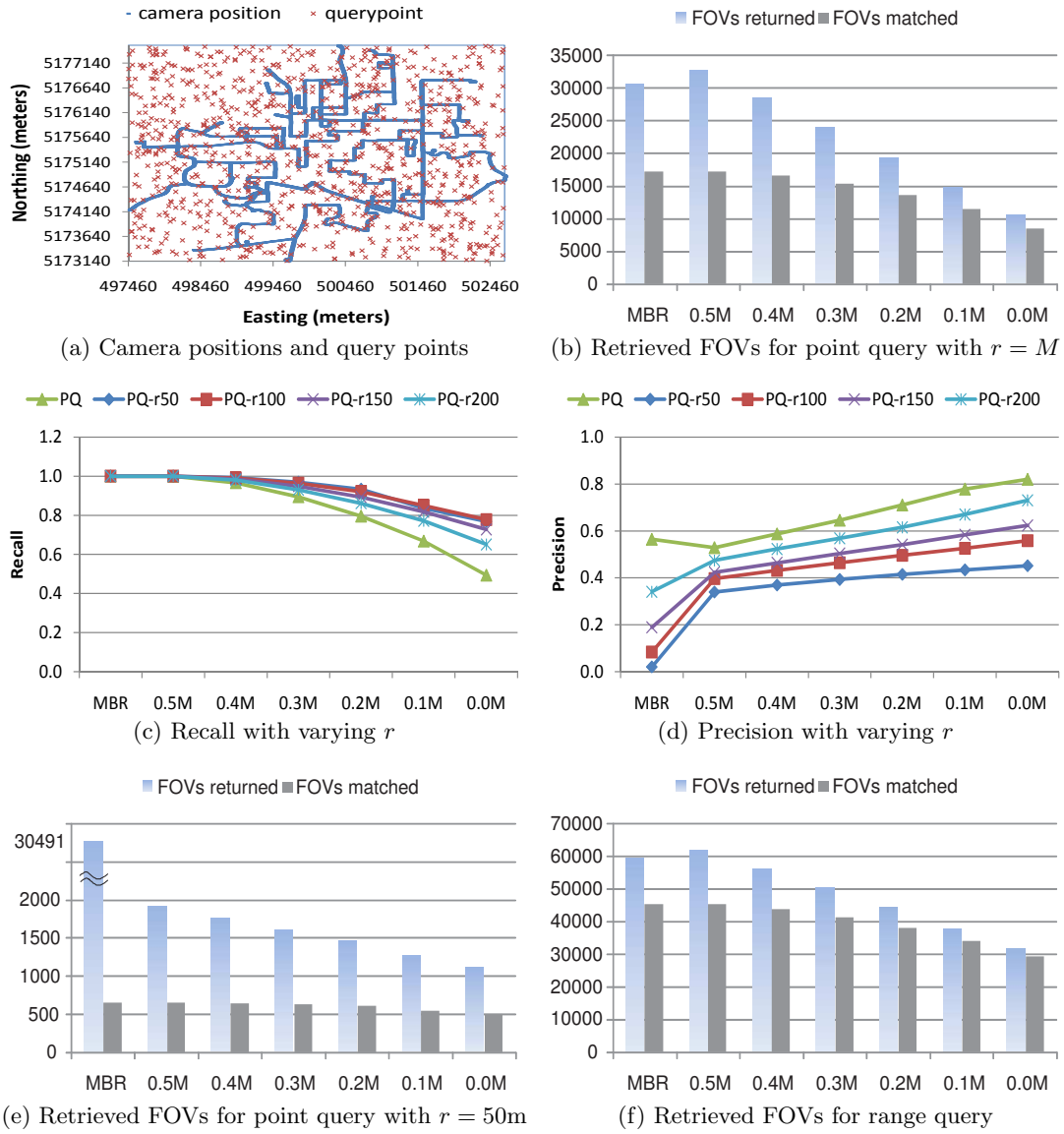
For all experiments we constructed a local MySQL database that stored all the FOV meta-data and their approximations (both MBRs and vectors). We used MySQL Server 5.1 installed on a 2.33 GHz Intel Core2 Duo Windows PC. For each query type described in Section 4 with the MBR and the vector approximation, we created a MySQL user defined function (UDF) to search through the FOVs in the database. We also implemented a UDF for the refinement step which returns the actual overlap instances between a query and an FOV. We used the Universal Traverse Mercator coordinates for all comparisons.

For the evaluation of the search results with different approaches, we computed the *recall* and *precision* metrics for the filter step. The recall is defined as the number of overlapping FOVs found in the filter step by an approach over the actual number of overlapping FOVs. Note that the actual number of overlapping FOVs is obtained after the refinement step from the exact geometric calculation using the circular sectors. The precision is defined as the number of actually overlapping FOVs found over the total number of FOVs returned in the filter step.

6.2 Comparison

We set the distance M to the maximum viewable distance among all recorded R values of FOVs, so M equaled 259 meters. The widest camera angle recorded was 60°. Thus, in all experiments we set the maximum overestimation constant to $\sin 30^\circ \times 259$, i.e., $0.5M$.

Point query: After executing 1,000 random point queries with 10,652 FOVs in the database, the number of actual overlap instances between a query point and an FOV was 17,203. This number was obtained and verified by geometric computation after the refinement step, i.e., it represents the ground truth. The point query results from the MySQL implementation are summarized in Table 1 and Figure 11.(b). The MBR approach returned 30,491 potentially overlapping FOVs in the filter step and found all 17,203 actually overlapping FOVs at the refinement step. The vector model was applied with varying δ values. As expected, with the maximum overestimation constant $\delta = 0.5M$ the vector model showed almost identical results to those of the MBR model (the size of the approximation is slightly bigger than with an MBR). However, when we decreased the value of δ, the vector model returned a smaller number of actually overlapping FOVs as well as a smaller number of potentially overlapping



(a) Camera positions and query points

(b) Retrieved FOVs for point query with $r = M$

(c) Recall with varying r

(d) Precision with varying r

(e) Retrieved FOVs for point query with $r = 50$m

(f) Retrieved FOVs for range query

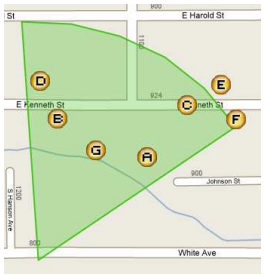
Figure 11: Summary of experimental results.

FOVs at the filter step. This is because the vector model is discriminating more against overlapping objects at the side of scenes as the value of δ decreases. Figure 12 provides an example of how different approaches perform the filter step. The MBR for the 42^{nd} FOV of video 61 overlapped with 7 query points while only 6 points actually overlapped with the FOV. The vector model found different numbers of query points as δ varied. As shown in Figure 12.(a), query points A, B, and G were located closer to the center vector of the FOV, so they were found in all approaches, even when $\delta = 0.0M$. However, D and F were very far from the center vector so they were only found when δ became larger. The vector model with a reduced δ found a smaller number of FOVs. The query points closer to the sides (i.e., those that may not be well perceived by humans) were effectively excluded. Overall, when δ grows the recall increases and the precision decreases.

We measured the time to execute 1,000 point queries with MySQL using various approaches. The bottom row of Table 1 shows the total amount of time in seconds reported by MySQL. On average, the vector models took 14-19% more time than the MBR model. We did not use indexes in the search so the results reflected the computational time for table scans. In reality, indexes such as B-trees or R-trees are used for a more efficient spatial search for a larger set of data and the execution time of the filter step depends on the performance of the indexes. Note that the reported time is for the filter step since the refinement step was implemented as a separate program with the results from MySQL. The focus of this study is not on the speedup of the filter step itself but on the overall query processing, and the effectiveness of the filter step to support versatile search using the characteristics of video. Even though the execution time of the vector-based filter step is a little longer than that of the MBR-based one, the number of selected objects from the vector model can be far smaller than that of the MBR model as shown in Tables 1, 2 and 3, which results in a significant speed up on the overall query processing by minimizing the

Table 1: Detailed results of point query.

	MBR	Vector (with different overestimation constant)					
		0.5M	0.4M	0.3M	0.2M	0.1M	0.0M
FOVs returned	30491	32535	28302	23843	19268	14762	10360
FOVs actually matched	17203	17197	16620	15390	13686	11488	8493
Recall	1.00	1.00	0.966	0.895	0.796	0.668	0.494
Precision	0.564	0.529	0.587	0.645	0.710	0.778	0.820
Exec. time of 1000 queries (sec)	8.5	10.5	10.5	10.0	10.0	9.7	9.7



(a) Query points on map

Label	Query ID	MBR	Actual	Vector (with different overestimation constant)					
				0.0M	0.1M	0.2M	0.3M	0.4M	0.5M
A	63	✓	✓	✓	✓	✓	✓	✓	✓
B	185	✓	✓	✓	✓	✓	✓	✓	✓
C	317	✓	✓	–	–	✓	✓	✓	✓
D	394	✓	✓	–	–	–	–	–	✓
E	465	✓	–	–	–	–	–	–	✓
F	740	✓	✓	–	–	–	–	✓	✓
G	761	✓	✓	✓	✓	✓	✓	✓	✓

✓: found, –: not found

(b) Query point overlaps using different approaches

Figure 12: Query points overlapped with an FOV (video 61, FOV 42).

workload of time-consuming refinement step.

Point query with bounding distance r: Figure 11.(e) shows the results of point queries with a bounding distance r between the camera position and the query point. When r was 50 meters the number of matching FOVs for 1,000 queries was 649. Note that 50m is approximately one fifth of the maximum viewable distance, which means that overlapping query points should be contained in 1/25 of the original FOV size. Thus, the number of overlap instances is greatly reduced. The MBR model returned the same 30,491 FOVs but, for example, the vector model with $\delta = 0.5M$ returned only 1,908 (a 94% reduction) with 1.0 recall. As δ decreased the recall diminished as well and the precision increased. This trend is analogous to the one observed on the results of point queries without a bounding distance. We repeated the same experiments while varying r from 50m to 200m. The results all exhibited the same trend as shown in Figures 11.(c) and 11.(d).

Figure 13 illustrates the effects of the r value on the search. When we searched for query point F (the Pizza Hut building in the scenes) without r (i.e., $r = M$), both frames shown in Figures 13.(a) and (c) were returned because they contain the query point. However, the building appears very small (and is difficult to be recognized by humans) in Figure 13.(c) since it was located far from the camera. Note that the same building is easily recognizable in (a) when the camera was closer to the object. We can effectively exclude (c) using an appropriate r value. Figures 13.(b) and (d) show the alternative satellite images of (a) and (c), respectively.

Directional point query: Using the same 1,000 query points, we searched for all FOVs that overlap with the query points while varying the viewing direction from the camera to the query point. We used a $\pm 5^\circ$ error margin with the viewing direction in all experiments. Table 2 shows the results of point queries with a 45° viewing direction. The MBR approach has no information about the direction in its estimation so it resulted in the same number of 30,491 FOVs which must be processed in the refinement step. When the overestimation constant is not too small ($\delta \geq 0.3M$), the vector model resulted in an approximately 90% reduction in the number of selected FOVs in the filter step compared to the MBR method, while providing a recall value of over 0.9. Significantly – as shown in Figure 12 – the missing FOVs mostly contained query points at the far sides of the viewable scene area. For different viewing directions, similar results were observed.

Table 2: Results of directional point query with $45^\circ \pm 5^\circ$. The actual number of matched FOVs were 402.

	MBR	Vector		
		0.5M	0.3M	0.1M
FOVs returned	30491	3858	2972	720
FOVs actually matched	402	389	381	134
Recall	1.000	0.968	0.948	0.333
Precision	0.013	0.101	0.128	0.186

Directional point query with bounding distance r: Table 3 shows the results of a very specific point query case, i.e., one that considers both the viewing direction and bounding distance. The vector model effectively excludes non-overlapping FOVs in the filter step. For example, with a 45° viewing direction and $r = 50m$ there were only 13 overlapping instances, which is a very small number with respect to the 1,000 queries and 10,652 FOVs. The vector model returned 374 FOVs, including the matched 13. Note that the MBR model returned 30,491 FOVs. We repeated the same experiments while varying δ and observed that the vector model provided the best balance between recall and precision with a value of $\delta = 0.3M$.

Table 3: Results of directional point query with r. $45^\circ \pm 5^\circ$ viewing direction and $\delta = 0.3M$.

	Vector			
	r=50	r=100	r=150	r=200
FOVs returned	374	1006	2124	2972
FOVs actually matched	13	93	151	264
Recall	1.000	1.000	1.000	1.000
Precision	0.035	0.092	0.071	0.089

(a) (b) (c) (d)

Figure 13: Impacts of bounding distance in video search.

Range query: We generated 1,000 random queries with an identical query region size of 100m by 100m, but different locations. For each range query, we checked the overlap between the query area and the FOVs. Figure 11.(f) summarizes the results, which show a similar trend as observed with point queries. The vector model with $\delta = 0.5M$ provided almost perfect recall, namely 0.998, with a slightly higher number of FOVs returned in the filter step. As δ diminishes the recall decreases and the precision increases. The chances for overlap between a given query range and any FOV increases as the size of the query range becomes larger. When we increased the query size to 300m by 300m, the recall of all approaches (even with $\delta = 0.0M$) became 1.0. At the same time, as the size of an approximation becomes larger, the number of false positives rises. When the size of the query range becomes smaller, the results approach those of the point queries in Table 1.

6.3 Illustration of Directional Query Results: A Real-world Example

We developed a web-based search system to demonstrate the feasibility and applicability of our concept of georeferenced video search (*http://eiger.ddns.comp.nus.edu.sg/geo/*). The search engine currently supports both directional and non-directional spatial range queries. A map-based query interface allows users to visually draw the query region and indicate the direction. The results of a query contain a list of the overlapping video segments. For each returned video segment, we display the corresponding FOVs on the map, and during video playback we highlight the FOV region whose timecode is closest to the current video frame.

In Figure 14 we illustrate an example of a directional query applied to our real-world georeferenced video data. We would like to retrieve the video segments that overlap with the given rectangular region while the camera was pointing in the *North* direction. Figure 14.(a) shows the video segments returned from the filter step using the MBR model. Recall that the MBR model retains no notion of directionality. Figure 14.(b) shows the results of the filter step using the vector model with input direction 0° (*i.e., North*) and $\delta = 0.3M$. Figures 14.(c) and (d) show the results from the refinement step with error margins $\pm 5°$ and $\pm 25°$ with respect to the given direction 0°, respectively. Note that we applied the refinement step on the output of MBR-based filter step shown in Figure 14.(a).

Using the MBR model the filter step returns videos with an aggregated duration of 775 seconds whereas the vector model based filter step returns only 98 seconds of videos. The refinement steps shown in Figures 14.(c) and (d) return 9 seconds and 65 seconds long videos for viewing directions $0° \pm 5°$ and $0° \pm 25°$, respectively.

The Figures 14.(a) through (d) also display the FOV visualizations for the corresponding video segments on the map. Note that a red FOV represents the frame currently being played in the video. As described in Section 3.1, the vector model is introduced as a fast filter step to quickly dismiss the unrelated videos and video segments. Figure 14 illustrates an example query where the vector model successfully eliminates most of the unrelated video segments, minimizing the amount of refinement processing.

7. CONCLUSIONS

In this study we proposed a novel vector-based estimation model of a camera viewable scene area. Our experimental results show that the vector model can be used in various ways to enhance the effectiveness of a search filter step so that the expensive and complex refinement step can be performed with far fewer potentially overlapping FOVs. The vector model successfully supports new geospatial video query features such as a directional point query with a given viewing direction from camera to object and a point query with a bounded distance between camera and object. We also demonstrated an immediate applicability of our proposed model to a common database by collecting real video data, implementing an actual database and queries using MySQL, and performing extensive experiments with the database.

So far we have focused on understanding the feasibility of new query features in georeferenced video search using the proposed model. We did not investigate query optimization issues that can impact the performance of the filter step, such as the most appropriate indexing structure. We are investigating query optimization as part of our ongoing work.

8. REFERENCES

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

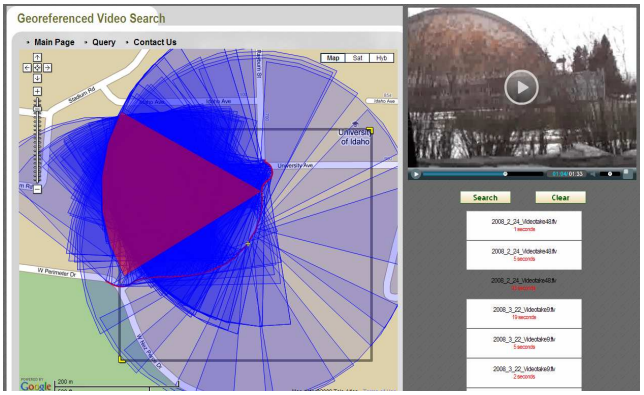
[2] *Woophy. http://www.woophy.com.*

[3] Sakire Arslan Ay, Roger Zimmermann, and Seon Ho Kim. Viewable Scene Modeling for Geospatial Video Search. In *ACM International Conference on Multimedia*, pages 309–318, 2008.

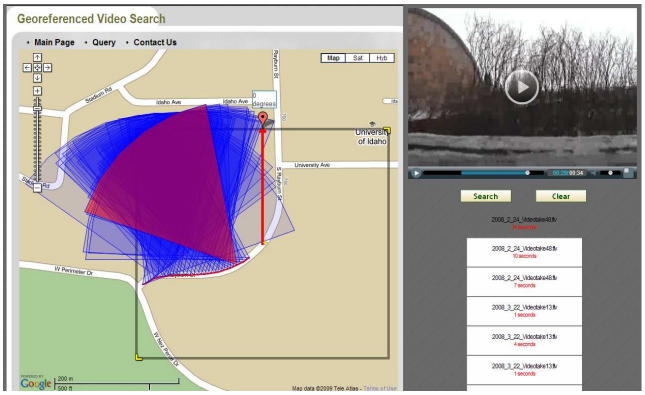
[4] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *ACM International Conference on Management of Data*, 1990.

[5] T. Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger. Multi-step Processing of Spatial Joins. In *ACM International Conference on Management of Data*, 1994.

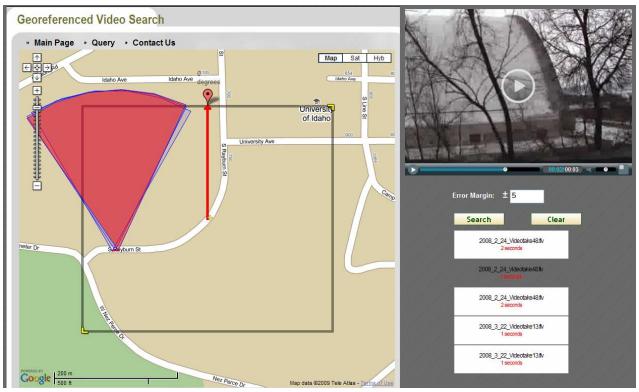
[6] Boris Epshtein, Eyal Ofek, Yonatan Wexler, and Pusheng Zhang. Hierarchical Photo Organization



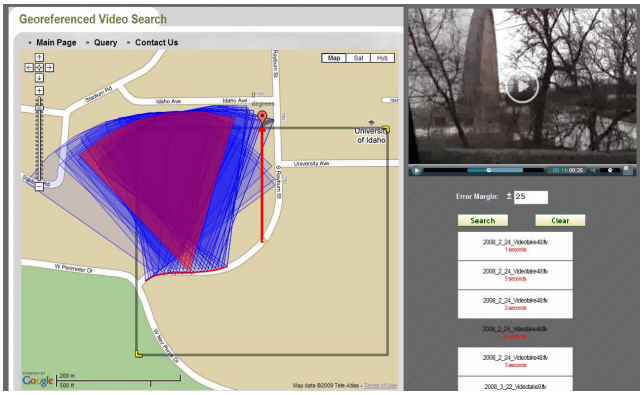
(a) Results of filter step using MBR model (no direction)



(b) Results of filter step using vector model (viewing direction=0° and $\delta = 0.3M$)



(c) Results of refinement step (viewing direction $0° \pm 5°$)



(d) Results of refinement step (viewing direction $0° \pm 25°$)

Figure 14: Illustration of directional range query results. The FOV of the current frame is highlighted.

Using Geo-Relevance. In *ACM Intl. Symposium on Advances in Geographic Information Systems*, pages 1–7, 2007.

[7] Shantanu Gautam, Gabi Sarkis, Edwin Tjandranegara, Evan Zelkowitz, Yung-Hsiang Lu, and Edward J. Delp. Multimedia for Mobile Environment: Image Enhanced Navigation. volume SPIE 6073, pages 1–11, 2006.

[8] 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.

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

[10] Lyndon S. Kennedy and Mor Naaman. Generating Diverse and Representative Image Search Results for Landmarks. In *International Conference on the World Wide Web*, pages 297–306, 2008.

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

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

[13] A. Orenstein. Spatial Query Processing in an Object-Oriented Database System. In *ACM International Conference on Management of Data*, pages 326 – 336, 1986.

[14] A. Pigeau and M. Gelgon. Building and Tracking Hierarchical Geographical & Temporal Partitions for Image Collection Management on Mobile Devices. In *ACM International Conference on Multimedia*, pages 141 – 150, 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] Ian Simon and Steven M. Seitz. Scene Segmentation Using the Wisdom of Crowds. In *Proc. ECCV*, pages 541–553, 2008.

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

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