Scan Without a Glance: Towards Content-Free Crowd-Sourced Mobile Video Retrieval System

Cihang Liu
School of Software
Tsinghua University
Beijing, China
cihang@greenorbs.org

Lan Zhang
School of Software
Tsinghua University
Beijing, China
zhanglan03@gmail.com

Kebin Liu
School of Software
Tsinghua University
Beijing, China
kebin@greenorbs.com

Yunhao Liu
School of Software
Tsinghua University
Beijing, China
yunhao@greenorbs.com

***Abstract*—Mobile videos contain rich information which could be utilized for various applications, like criminal investigation and scene reconstruction. Today's crowd-sourced mobile video retrieval systems are built on video content comparison, and their wide adoption has been hindered by onerous computation of CV algorithms and redundant networking traffic of the video transmission. In this work, we propose to leverage *Field of View(FoV)* as a content-free descriptor to measure video similarity with little accuracy loss. Based on FoV, our system can filter out unmatched videos before any content analysis and video transmission, which dramatically cuts down the computation and communication cost for crowd-sourced mobile video retrieval. Moreover, we design a video segmentation algorithm and an R-Tree based indexing structure to further reduce the networking traffic for mobile clients and potentiate the efficiency for the cloud server. We implement a prototype system and evaluate it from different aspects. The results show that FoV descriptors are much smaller and significantly faster to extract and match compared to content descriptors, while the FoV based similarity measurement achieves comparable search accuracy with the content-based method. Our evaluation also shows that the proposed retrieval scheme is scalable with data size and can response in less than 100ms when the data set has tens of thousands of video segments, and the networking traffic between the client and the server is negligible.**

***Keywords*-crowd-sourced; mobile video retrieval; Field of View**

I. Introduction

With the fast development of smart devices, numerous on-board cameras lead to ubiquitous video sources. According to International Telecommunication Union, by May 2014, there are nearly 7 billion mobile subscriptions worldwide, which is equivalent to 95.5% of the world population. In the United States, over 50% of the mobile audience take photos with mobile phones and over 20% of them record mobile videos. Video data are rich in visual/non-visual information, and taking good use of massive video data will bring us great benefits. One of the most important applications is content-based video retrieval. Existing work designed different types of descriptors to characterize features of video content [1]. By measuring similarity between content descriptors, video retrieval systems can automatically detect desired targets in videos, *e.g.*, human faces [2], crime behaviors [3] and security events [4]. Crowd-sourced videos can greatly increase the power of video data. For example, in 2013, the Boston bombing causes three deaths and 150 injuries. After the attack, with the help of the videos taken by the thousands of attendees to the event, the police successfully pinpoint the suspects, scout for additional bombs, and figure out who was at each scene within five days.

There is no doubt that the crowd-sourced video retrieval system is a must in the era of big data. Existing video retrieval solutions ([5] [6] [7]) inevitably fall into two categories, data-centric and query-centric. Data-centric solutions rely on clients uploading their mobile videos to the data center. The powerful data center will do the video content computation and client users only need to transmit query request and receive search result. In query-centric systems, cloud server only distributes queries. When the clients receive the queries, they perform the content retrieval algorithm locally and return the results to the cloud. However, using existing content based retrieval techniques, neither of the architectures is practical in a crowd-sourced video scenario. Firstly, as the data volume of the crowd-sourced videos is extremely large, uploading video data via cellular network will be extremely time-consuming and money-consuming. Secondly, mobile videos contain a lot of personal information. Uploading raw video data to the server may incur users' privacy concerns. Thirdly, video data are unstructured high-dimensional data. CV algorithms for video content analysis and comparison are computation-intensive. It is not efficient to conduct heavy CV algorithms over crowd-sourced videos on either mobile devices or data centers.

We notice that, most overhead of a content based retrieval system is caused by transmitting and analyzing unmatched videos. To achieve efficient crowd-sourced videos retrieval, it is essential to cut down unnecessary transmission and computation. The dilemma compels us to recall the motivation of crowd-sourced videos retrieval, that is the user tends to take use of ubiquitous video cameras to learn ”When”, ”Where”, ”Who” and ”What”. Since ”Who” and ”What” requires expensive video content analysis, we can use ”When” and ”Where” to filter out unwanted video clips. In other words, rather than analyzing video content directly as current video retrieval systems, we propose to significantly narrow down videos by ”Where” and ”When”, which means first searching videos in a certain area during a specific time interval. Based on this idea, in this work we propose a content-free crowd-sourced videos retrieval system. As the name implies, rather than the visual content, the content-free system tends to formalize the spatio-temporal information embedded in mobile videos, which could be managed at a much lower cost.

Previous works have laid a good foundation for content-free video systems. For example, [8] proposed a model for mobile video search. The viewable scene of each video frame is defined as a tuple (p, θ, r), in which the elements represent the location, orientation and radius of view of the viewable sector. [9] further builds a GeoTree. Adjacent video frames can be aggregated and the bounding rectangle of these frames is constructed and indexed in the tree. However, there are still challenges when constructing a mobile video retrieval system. Firstly, the search should be conducted on both temporal and spatial dimensions, but none of the existing work considers the temporal information of videos. Existing architecture only return a set of discrete video frames whose scenes have intersection with the query range, rather than continuous video segments. Secondly, the existing aggregation rules of video frames are simple and far from practical. Those rules require the user to shift the camera in the same direction with the orientation of the device, or rotate the camera at a constant spot. But in real life, the movements of the user are complicated and unpredictable.

Facing these challenges, we design and implement a practical content-free mobile video system. By defining the content-free video descriptor FoV(Field of View) [8], videos are segmented into clips in real-time and descriptors are uploaded to the cloud when the subscriber finishes recording. We design the indexing structure maintained at the cloud side to guarantee high efficiency of video search. Our main contributions are summarized as follows:

- We propose a new content-free similarity measurement over mobile video frames, which achieves high efficiency and comparable matching results with the content based method. This method imposes no constraint on device subscribers' movements, and works in complex practical scenarios.

- We design and implement two real-time algorithms to extract the content-free feature of the video frames and segment the video into clips according to our similarity measurement. Both the computation cost for feature extracting and networking traffic for feature uploading are extremely small.

- We propose an efficient indexing structure that takes both spatial and temporal information of the content-free feature into consideration. With the index structure maintained at the server side, a user can search top-k most relevant video segments with a minimized response time.

The rest of this paper is organized as follows. Section II specifies the problem definition and system overview. Section III and section IV discuss our FoV similarity measurement and video segmentation schemes respectively. In section V, we describe the video retrieval and rank algorithms. The system implementation and evaluation are specified in section VI. Section VII discusses some design issues. Section VIII discusses existing efforts that are related to this work and we conclude our work in section IX.

II. Problem Definition and System Overview

In general, there are three parties involved in a crowd-sourced video retrieval system. On the client side, a *video provider* records videos using his/her smart devices (*e.g.*, smart phones and smart glasses) and would like to contribute them to the retrieval system. Videos could be taken while the providers are moving, *e.g.* walking and driving. A *querier* need to retrieve video clips whose content satisfy some specific temporal and spatial criteria. The *cloud server* collects description information of videos from providers, such as spatio-temporal information and feature descriptors, and searches target videos for the querier. To speed up the search procedure while remaining the search accuracy, the server can index the video information. When the server receives a query, it looks up the index and return the most relevant video to the querier.

A. Problems

While designing an efficient and accurate FoV based crowd-sourced mobile video retrieval system, we must solve the following questions:

1. *Rational similarity measurement over FoV*

FoV used to be a concept about videos with stable locations and orientations in previous research findings [8] [10]. In reality, the FoV of the video will also be dynamic since location and orientation of the camera cannot be a constant value. Thus, there must be a rational measurement to help us distinguish between different FoVs.

2. *Sound segmentation algorithm for mobile video*

In a piece of mobile video, FoV may keep changing along with the user's movement. When an inquirer activates a search request, it is uneconomical to upload the whole video if there is few useful FoVs. Therefore, an FoV similarity based video segmentation algorithm is needed for both the server side and the client side. On one hand, the server can set up the index according to different FoVs. On the other hand, uploading the relevant video segment targeted to the query can save a lot of web traffic.

3.*Efficiency on the client side*

The preprocessing of the video needs to be performed on the client side as it is never economical to upload the whole video after capturing. However, performing traditional CV algorithms on the client side is not applicable since smart phones are resource limited as always. Efficiency must be taken into account in algorithm design and system implementation.

4.*Time-sensitive index over FoVs*

Existing indexing schemes on FoVs only consider the stable location information of the FoV [10], which can be easily solved by the sophisticated spatial indexing technology. In real situations, FoV of the camera keeps changing their location along with the users' movement. In other words, both video clips and FoVs are always time-sensitive. The query of an inquirer may also contain a specific time interval. How to efficiently index the spatio-temporal information of all the FoVs is still unsolved by the previous work.

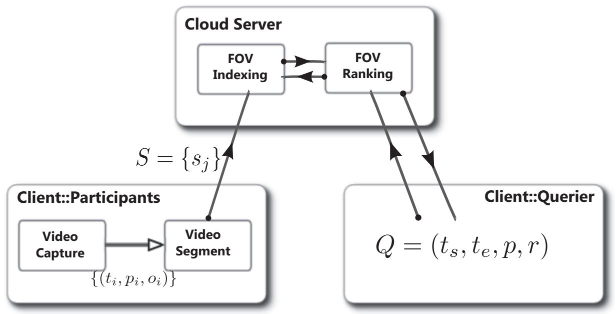


Fig. 1. System Architecture

B. FoV

Generally speaking, we use video frames to describe the content of a continuous video. A video frame is a single picture that is shown as part of a larger video. Each video consists of a sequence of video frames. Traditional descriptors all focus on the visual feature of the video frames and try to use matrixes or vectors to describe the video content. However, both matrix and vector based video features are not applicable for crowdsourcing systems, since the extraction and similarity measurement are time-consuming.

To address this question, we recall the details when recording a video clip. By common sense, we know that when a subscriber is capturing videos with their smart phones, the camera will cover an visible area. Meanwhile, according to the principle of optical imaging, this area is a conical space and it will change along with the movement of the user. FoV(Field of View), as its name, ably helps to describe this area. It is a video descriptor which focuses on the field of view of each video frame. Compared with the traditional descriptors, negligible data size and low dimension are the major characteristics of FoV.

In our work, FoV is defined in the form of 2-tuple in order to describe the vertex and vertex angle of the conical area:

$$f = (p, \theta), \tag{1}$$

where p stands for the GPS location, consisting of $p.lat$(latitude) and $p.lng$(longitude), and θ stands for the azimuth angle of camera.

Moreover, every camera is born with a fixed viewing angle $\mathscr{A} = 2\alpha$. Thus, the angle range covered by the camera is $\Theta = (\theta - \alpha, \theta + \alpha)$.

C. System Overview

With the help of FoV, we propose an efficient crowd-sourced mobile video retrieval system. As Figure 1 illustrates, there are four major components in our system: (1) video capturing; (2) video segmentation and representative FoV extraction; (3) representative FoV indexing; (4) rank based retrieval. Here is a brief overview about the system workflow.

When a user is capturing a video with his/her smart phone, the orientation and location information of the device will be synchronously collected at the backstage. Meanwhile, the application merge these two kinds of information with the timestamp to the form of (t_i, p_i, θ_i), where t_i, p_i and o_i represent the timestamp, location and azimuth angle corresponding to the i^{th} frame. And this data record will directly passed to the module of video segmentation on the client side.

A real-time video segmentation is implemented in the segmentation module. It absorbs the incoming data record and make a decision on whether to segment the video in O(1) time complexity. If so, the start time and the end time of the current video segment will be recorded and the representative FoV (including the time interval and representative location and azimuth orientation) will also be obtained in real time.

After the user stop video capturing, the segmentation module will end at the same time, and we can get a set of FoVs each of which represents a video segment. The set of FoV will be uploaded to the cloud server.

The server maintains a dynamic index structure for the upload FoVs. In the structure, time interval and location information are considered so that the retrieval efficiency can be guaranteed.

An inquirer may activate a search request with a query $Q = (t_s, t_e, p, r)$, which indicates that he/she wants to search all the videos that covers the circular area, with p as the center and r as the radius, from start time t_s to end time t_e. The server will rapidly look through the index and return the inquirer a list of search results. The elements in the list are sorted in descending of relevance according to the query Q.

So far, we have given a brief introduction for our system architecture and workflow. More details of each module will be elaborated in the following sections.

III. Similarity Measurement

While clients start recording videos via smart devices, a background process is activated to track the FoV of each video frame and determine when to segment video according to the correlation of FoVs. In this section, an efficient and rational similarity measurement over FoV will be introduced. Under this similarity measurement, any pair of FoVs can be linked together via a two-phase transformation. Compared to the traditional vector based similarity(distance) measurement, FoV based similarity measurement brings incredible high efficiency in calculation and low cost in storage.

A. Similarity Measurement

Similarity measurement is a real-valued function that quantifies the similarity between two objects. In the field of information retrieval, most of the existing similarity(distance) measurements are based on feature vectors/matrixes. For example, when we want to get the Euclidean distance of two vectors, it is essential to go through each dimension of the feature vector and get the deviation of them. However, with the proposed measurement, the similarity between two FoVs can be obtained with a simple mathematical calculation, which is far more lightweight than ordinary algorithms. In addition

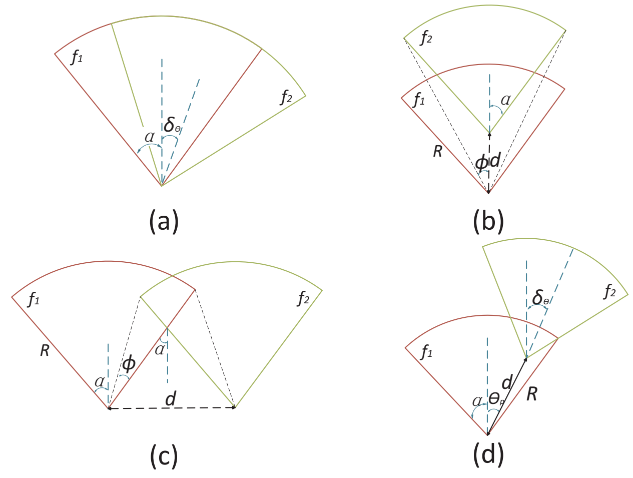


Fig. 2. Schematic View

to the high efficiency, the evaluation also verify the rationality of our algorithm.

Preliminaries:

f_1 and f_2 denote two FoVs which $f_1 = (p_1, \theta_1)$ and $f_2 = (p_2, \theta_2)$ with their respective angle range Θ_1 and Θ_2. δ_p denotes the distance between p_1 and p_2, and δ_θ denotes the orientation difference. Thus, we have

$$\delta_p = |\vec{p_1 p_2}|, \delta_\theta = \min(|\theta_2 - \theta_1|, 360 - |\theta_2 - \theta_1|) \quad (2)$$

Besides, in consideration of normalization, for $\forall f_1, f_2$

$$Sim(f_1, f_2) \leq 1, \quad (3)$$

Equation (3) holds if and only if $f_1 = f_2(\delta_p = 0$ and $\delta_\theta = 0)$, which indicates the two FoVs are the same. Apart from this, any increment in either δ_p or δ_θ will cause the reduction in $Sim(f_1, f_2)$.

As we all know, in Newtonian mechanics, any rigid body motion can be divided into translation and rotation. Based on this theory, while recording, we think of the motion of mobile devices as a translation in δ_p and a rotation in δ_θ. Under this proposition, any FoV can be associated to the other. In the following paragraphs, we will interpret these two circumstances separately and elaborate how they can be combined together.

Case 1: Rotation

When a rigid body is rotating, the barycenter of it keeps fixed and other parts of it move around the center. In this paper, the rotation situation is that the location of the camera stands still and the orientation of the device changes. As shown in Fig. 2(a), the rotation will cause the change of $\delta_p = 0$ and $\delta_\theta \neq 0$. Intuitively, we make the coverage intersection be the similarity between two FoVs.

$$Sim_R(f_1, f_2) = \frac{|\Theta_1 \cap \Theta_2|}{|\Theta|} = \begin{cases} \frac{2\alpha - \delta_\theta}{2\alpha} & \delta_\theta < 2\alpha \\ 0 & otherwise \end{cases} \quad (4)$$

Apparently, the intersection of f_1 and f_2 will decrease with the increase of δ_θ, which cause the linear decrease of similarity. When δ_θ comes to 2α and continues to rise up, we can find that there will be no intersection between the two sectors. In this circumstance, the similarity between the two FoVs will stay at 0 before the camera turns around.

Case 2: Translation

Translation is the movement that changes the position of an object, as opposed to rotation. A displacement is called a translation if a body is moved from one position to another and the lines joining the initial and final points of each of the points of the body are a set of parallel straight lines.

In this case, f_1 and f_2 maintains the same orientation θ, and f_2 can be considered as f_1 with a translation of distance δ_p. However, the translation direction θ_p varies between $0°$ to $360°$, and the translation direction will affect the descending rate of the similarity between them.

Without loss of generality, for arbitrary FoV pair f_1 and f_2, with the help of orthogonal decomposition, the translation from f_1 to f_2 can be divided into two steps, the parallel translation and vertical translation (shown in Fig. 2(b) and Fig. 2(c)).

Intuitively, the similarity of parallel translation, denoted as $Sim_\parallel$, decreases slower than that of vertical translation, denoted as $Sim_\perp$ Next, we will separately consider these similarity functions about these two extreme cases.

Fig. 2(b) illustrates the situation of parallel translation, where the translation of f_2 is in the same direction with θ. R represent the radius of the view we can see and d is the translation distance. According to the optical principles, relative to f_1, the viewing angle covered by f_2 is narrowed from 2α to 2ϕ, which can be obtained by a a few steps of triangular transformation:

$$\phi_\parallel = \arctan \frac{R \sin\alpha}{d + R\cos\alpha}, \quad (5)$$

Fig. 2(c) illustrates the circumstance where the translation of f_2 in the vertical direction of θ. The angle $\phi_\perp$ can be obtained by similar transformation:

$$\phi_\perp = \arctan \frac{\begin{bmatrix} 1 & 0 \end{bmatrix} AB}{\begin{bmatrix} 0 & 1 \end{bmatrix} AB}, \quad (6)$$

where $A = \begin{bmatrix} 2R\sin 2\alpha & -2\sin 2\alpha \\ 2R\cos 2\alpha & 1 - \cos 2\alpha \end{bmatrix}, B = \begin{bmatrix} \sin\alpha \\ d \end{bmatrix}$.

It is obvious that, under a same translation distance d, the angle reduction of $\phi_\parallel$ is bigger than $\phi_\perp$.With the value of $\phi_{\parallel(\perp)}$, the transformation similarity can be obtained via a unified equation:

$$Sim_{\parallel(\perp)} = \frac{\phi_{\parallel(\perp)}}{2\alpha} \quad (7)$$

.

According to the mathematical calculation, we have following statements about $Sim_\parallel$ and $Sim_\perp$:

1.For the same radius R and translation distance d, the following inequality always holds and the equality holds if and only if $d = 0$

$$Sim_{\|} \geq Sim_{\perp} \tag{8}$$

2.With the increase in translation distance d, $\phi_{\|(\perp)}$ will shrink from $\Theta = 2\alpha$, which will result in the decrease of similarity. However, $Sim_{\|}$ will always be a positive real number but $Sim_{\perp}$ will drop to 0 when d reaches $2R\sin\alpha$.

To sum up, $Sim_{\|}$ and $Sim_{\perp}$ represent the minimum and maximum of the decreasing gradient of similarity. With the propositions above, any translation of video cameras can be divided into two steps. For example, when the translation of f_2 is in the direction of $\theta_p \in [0°, 90°]$ (see in Fig.2(d)) with a translation distance d, we can have the translation similarity by a weighted equation:

$$Sim_T(f_1, f_2) = (1 - \frac{\theta_p}{90})Sim_{\|} + \frac{\theta_p}{90}Sim_{\perp} \tag{9}$$

Case 3:Reality

In real life, any rigid movement can be seen as the combination of rotation and translation. Newtonian mechanics inspires us separately consider the rotation and translation of FoVs, and combine them together. Under this inference, for an arbitrary FoV pair $f_1 = \{p_1, \theta_1\}$ and $f_2 = \{p_2, \theta_2\}$, as shown in Fig. 2(d), f_2 can be considered as a transformation, which consists of a rotation of f_1 in δ_θ and a translation with a distance δ_p in direction of θ_p. The similarity between f_1 and f_2 can be obtained by Equation (10).

$$Sim(f_1, f_2) = Sim_R \times Sim_T \tag{10}$$

In later chapters we will evaluate our theorem by comparing FoV based measurement and computer vision based measurement.

IV. Video Segmentation

Video retrieval systems absorb users' preference and return the relevant video clips. As what users want are always piecemeal and originate from different videos, segmenting vides into segments according to their content will help the server set up an index structure for videos and retrieve what the users exactly want as efficient as possible. In this section, we propose a novel, FoV based, video segmentation algorithm that can be performed in real time. Meanwhile, representative FoVs of the segments will be extracted and uploaded to the server. The following paragraphs will show the details of these two algorithms.

A. Video Segmentation

When capturing videos with mobile phones, users will inevitably hold the cameras and keep walking around. which will result in the changes of FoV. In crowd-sourced video retrieval systems, inquirers always want to fetch all the videos that cover a specific location during a specific time interval. This demand exactly corresponds with the definition of FoV.

In this case, FoV based mobile video segmentation algorithm can help to classify the video segments according to different fields of view. Since client users only need to upload the relevant video segment instead of the full video, With the optimization of the accuracy of video retrieval, the redundant web traffic can also be minimized.

Here is a formal description of the proposed segmentation algorithm, our purpose is to segment a continuous video into $S = \{s_1, s_2, ..., s_n\}$, where each element s_i represents a piece of video segment. Each s_i consists of several FoVs, a start time t_{si} and an end time t_{ei}.

As is discussed in the previous section, for any FoV pair f_1 and f_2, f_2 can be seen as the transformation of f_1 with a rotation in δ_θ and a translation in δ_p. Based on this hypothesis, Algorithm 1 shows the pseudo code of the segmentation algorithm.

Algorithm 1 Video Segmentation

```
Input: The FoV sequence of an video $F = \{f_1, f_2, ..., f_n\}$,
    segmentation threshold $thresh$
Output: The segment set $S = \{s_1, s_2, ..., s_k\}$, where $s_i = \{s_1, s_2, ..., s_{n_i}\}$
 1: $t_{s1} = 1, t_{e1} = 1, k = 1, S = \Phi, f_s = f1$
 2: //Time slot iterator
 3: for each $i \in [1, n]$ do
 4:     if $Sim(f_s, f_i) < thresh$ then
 5:         //Segment the video and add the segment into S
 6:         $t_{ek} = i - 1$
 7:         $S = S \cup \{s_k\}$
 8:         //Initialize a new segment
 9:         $k = k + 1, s_k = \{f_i\}, f_s = f_i$
10:         $t_{sk} = i, t_{ek} = i$
11:     else
12:         //Add $f_i$ into the current segment $s_k$
13:         $s_k = s_k \cup \{f_i\}$;
14:     end if
15:     if $i = n$ then $S = S \cup \{s_k\}$
16:     end if
17: end for
18: return $S$;
```

The input of the algorithm consists of a sequence of FoVs F and the segment threshold $thresh$. The algorithm iterates the time i and sequentially processes the corresponding FoV f_i. To get video segment s_k, it just focus on the initial FoV f_s and the current FoV f_i. If the change of the FoV has reached the similarity threshold ($Sim(f_s, f_i) < thresh$), the process will suspend and the current segment s_k will be added into the segment set S. Meanwhile, the algorithm will initial a new segment with f_i and continue with the segmentation process. When the iteration is finished, we can get the result of the segmentation $S = \{s_1, s_2, ..., s_k\}$.

B. Segment Abstraction

After the video segmentation, each segment s_i in the set S consists of several FoVs and FoVs in the same video segment have similar fields of view and continuous timestamps. To give segment s_i a further abstraction, we use the segment abstraction algorithm to extract the representative FoV f_{ri} . By doing this, the original video can be concentrated and represented by a set of representative FoVs. Users' web traffic of uploading will be minimized and the indexing structure on the cloud side will be more targeted.

To get the representative FoV f_{ri}, all the FoVs in s_i will be the input of the segment abstraction algorithm according to the Equation (11). The location and orientation of f_{ri} will be obtained by computing the arithmetic average of all the elements in segment s_i. In addition, t_{si} and t_{ei} will also be extracted and uploaded along with f_{ri}.

$$\begin{cases} \bar{p}_i = & \dfrac{\sum p}{|S_i|} \\ \bar{\theta}_i = & \dfrac{\sum \theta}{|S_i|} \end{cases} \tag{11}$$

C. Complexity Analysis

Intuitively speaking, both video segmentation algorithm and segment abstraction algorithm process the input data linearly and come up with the output. For video segmentation, the algorithm will go through the for-loop m times(m denotes the number of FoVs, which is equivalent to the number of video frames). Each loop will simply perform a similarity measurement between two FoVs and decide whether to initial a new video segment or not. On the other hand, computing the arithmetic average will also cause linear time complexity and no extra space complexity.

To sum up, for a set of FoVs S, the time complexity to perform video segmentation and segment abstraction are both $O(|S|)$, which is theoretically the lowest. Moreover, the experiment result expresses that both functions could be implemented as intent listeners in a real-time invocation environment.

V. Rank Based Video Retrieval

After capturing mobile videos, representative FoV of each video segment will also be extracted. As video contributors, users can choose to upload these representative FoVs for video retrieval. On the other hand, as video inquirers, when a video search request is activated, the server will look up in the index and return a series of relevant video segments. As there can be pervasive video contributors and search inquirers, it is necessary for the cloud server to maintain an efficient and dynamic index to provide the efficient updating and retrieval.

In this section, we propose an R-Tree based FoV indexing structure to index the representative FoVs uploaded from the client users. The index structure takes the spatio-temporal information of the representative FoVs into account so that it can satisfy the demand for retrieval efficiency and accuracy. When an inquirer performs a range based request, the server will return a rank based result, containing all the FoVs that have intersections with this range.

A. Setup of the FoV Index

R-tree is a height-balanced indexing structure to handle spatial data [11]. In R-tree, data objects are represented by intervals and they often cover areas in multi-dimensional spaces. It is the MBRs(Minimum Boundary Rectangles) of the data object that are stored and managed in the tree. So an R-Tree is good at answering range-based query. When searching in R-Trees with a query rectangle Q, the search algorithm descend from the root node and retrieve all the MBRs that intersects with Q. R-Tree based indexing scheme can be used to index the spatio-temporal information of the representative FoVs perfectly. In normal cases, an R-Tree is used to index multidimensional rectangles. Each rectangle is described as two double arrays $min[]$ and $max[]$, which represent the minimums and maximums of all dimensions. In FoV index, for each representative FoV $f_r = (\bar{p}, \bar{\theta})$ with segment start time t_s and end time t_e, we construct an FoV rectangle with $min[] = [\bar{p}._{Lng}, \bar{p}._{Lat}, t_s]$ and $max[] = [\bar{p}._{Lng}, \bar{p}._{Lat}, t_e]$ and insert it into the R-Tree. As we can see that both arrays share the same values in the first two dimensions, each FoV rectangle draws a segment in 3-d space.

B. Rank Based Retrieval

When performing an retrieval, the input will also be a rectangle R with $min[]$ and $max[]$, and the R-Tree will return all the rectangles that have intersection with R. In this paper, an inquirer will send a search request by $Q = (t_s, t_e, \hat{p}, \hat{r})$, where t_s and t_e represent the start time and the end time of the request interval and $\hat{p}$ and $\hat{r}$ represent the center position and the radius of the query range. The server converts $\hat{r}$ to the longitude scale $\hat{r}._{Lng}$ and latitude scale $\hat{r}._{Lat}$ corresponding to $\hat{p}$ and constructs an query rectangle $\hat{R}$ with the spatio-temporal information embedded in Q and perform a retrieval in the R-Tree, where $min[] = [\hat{p}._{Lng} - \hat{r}._{Lng}, \hat{p}._{Lat} - \hat{r}._{Lat}, t_s]$ and $max[] = [\hat{p}._{Lng} + \hat{r}._{Lng}, \hat{p}._{Lat} + \hat{r}._{Lat}, t_e]$. Then the server will return all the representative FoVs that have both spatial overlap and temporal overlap with $\hat{R}$.

However, traditional intersection based retrieving scheme need to be modified due to the following reasons.

First of all, intersections of MBRs don't make sense in FoV based retrieval systems. The spatial information in an FoV reveals the location of the device. However, inquirers never want to know where the cameras are. The only thing they care about is whether there is a video segment **covering** the query range they want.

Secondly, the scale of the query range is hard to decide. The value of the radius determines the size of the query rectangle we choose. There is a tradeoff between accuracy and efficiency. A smaller $\hat{r}$ will help to improve the search efficiency, but representative FoVs out of the query rectangle might be ignored. Similarly, the increase of $\hat{r}$ will also affect the above two metrics.

At last, besides the spatio-temporal information, the search mechanism must take the orientation information into account. Since smart phone cameras have a limited viewing angle, different orientations of the cameras will cause different FoVs even they share the same position. For example, if a user is capturing the video of Angela Dorothea Merkel on the grandstand, the video is still useless for an inquirer who wants to see the World Cup Final even the user is in the first row.

To deal with these challenges, we design a filtering mechanism based on the traditional retrieving scheme.

1.Based on the inquirer's configuration, to generate a reasonable query rectangle, an empirical radius of view is selected to form the query range, *e.g.* $20m$ for the residential area and $100m$ for the highways, .

2.Because there could be trees of walls obscuring our vision, closer FoVs will have a higher probability to cover the query area. As a result, FoVs retrieved from the R-Tree are sorted by the distance between the FoV and the circle center $\hat{p}$.

3.By iterating the retrieved representative FoVs, the server will exclude the FoVs that have the improper direction and rank the remaining by the distance to query point $\hat{p}$

4.According to the inquirer's requirement, the top N records among the results will be return to the inquirer.

VI. Implementation and Evaluation

A. Implementation

The prototype system can be divided into two parts, the client application and server program. The client application is implemented with android SDK 11 and the server side is implemented with Java1.6. Meanwhile, thanks $OPENCV$, for providing us with its comprehensive CV interfaces and algorithms.

Transformation of GPS Information

The location information obtained from the embedded GPS chip is in the form of (Lng, Lat), which represent the longitude and the latitude of the device. For a pair of FoVs $f_1 = (p_1, \theta_1)$ and $f_2 = (p_2, \theta_2)$,where $p_1 = (Lng_1, Lat_1)$ and $p_2 = (Lng_2, Lat_2)$, it needs a transformation to get the translation distance δ_p and translation orientation θ_p.

We consider the Earth as a regular sphere with the radius $r_e = 6378140$m. The longitude and latitude of the earth are evenly divided into 360 parts respectively. Since the space covered by an FoV is relatively small to the size of the Earth, we consider δ_p and θ_p to be in the Euclidean space.

The equation of the components are as follows:

$$\begin{cases} \delta_{p_x} = & \dfrac{2\pi r_e \cos\frac{Lng_2 - Lng_1}{2}}{360}(Lng_2 - Lng_1) \\ \delta_{p_y} = & \dfrac{2\pi r_e (Lat_2 - Lat_1)}{360} \\ \delta_p = & \sqrt{{\theta_{p_x}}^2 + {\theta_{p_y}}^2} \\ \theta_p = & \arctan\dfrac{\delta_{p_x}}{\delta_{p_y}} \end{cases} \tag{12}$$

Clock Synchronization

In our system, crowd-sourced videos are captured with different mobile devices by different subscribers. In most situations, every device has its own local clock and clocks of different devices, as well as the server, are usually unsynchronized. However, all the COTS devices support clock synchronization service via the satellites. Devices can easily have subsecond time deviations from the global clock with NTP/SNTP. Meanwhile, video retrieval systems are not sensitive to time deviation because a video of several milliseconds will make negligible difference to users' cognition. Thus, we don't need to perform a supernumerary clock synchronization between clients and the server.

B. Evaluation

We use off-the-shelf devices to evaluate the performance of our system from different aspects. The client application is implemented on HTC New One, with 1.7GHz quad processor and 2GB RAM. The server side is implemented on Macbook Air MD761.

As is proposed before, for any FoV pair f_1 and f_2, f_2 can be seen as a 2-phase transformation of f_1, which consists of a rotation of δ_θ and a translation in distance of δ_p and orientation of θ_p. The similarity of f_1 and f_2 can be obtained by Sim_R multiplied by Sim_T. And Sim_T is also based on two components $Sim_{||}$ and $Sim_\perp$.

As previously described, rotation and translation will both affect the similarity between two FoVs. The decrease of Sim_R is proportional to the angle δ_θ, until Sim_R falls to 0. On the other hand, the decrease of Sim_T is relevant to the translation distance δ_p and translation orientation θ_p, as well as the radius of view R. Fig. 3 illustrates the theoretical value of the two extreme cases, $Sim_{||}$(above) and $Sim_\perp$(below), where $\theta_p = 0°$ and $\theta_p = 90°$.

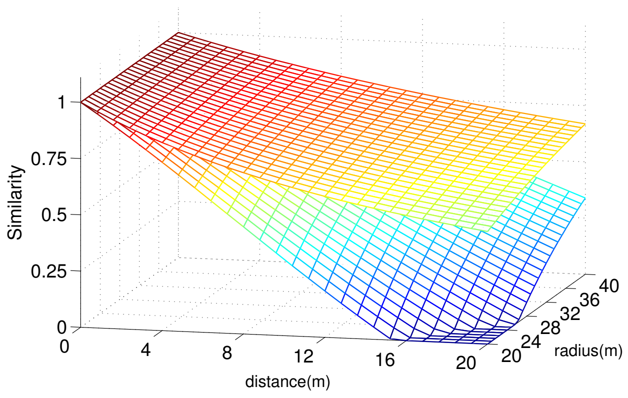
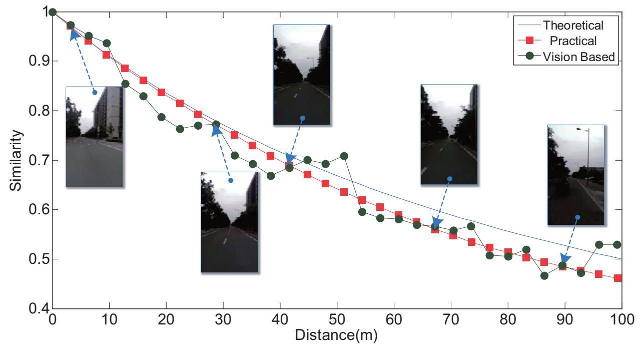


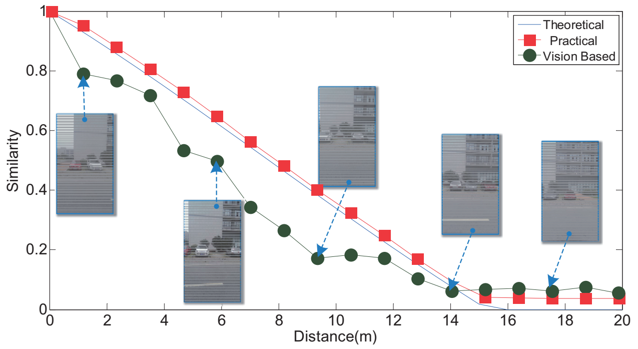
Fig. 3. Translation Similarity Model

1) Similarity Measurement: To prove the rationality of the model, we perform two kinds of experiments. We capture two kinds of videos when walking down the street, with $\theta_p = 0°$ and $\theta_p = 90°$, which correspond to the formal two circumstances.

Fig. 4 plots the comparison between the theory value and practice value. In each subfigure, the blue line indicates the theoretical value of $Sim_{||}$ (or $Sim_\perp$)and the the red line indicates the calculation result by the collected data from the smart phone sensors. Then we use frame differencing algorithm(as a representative of CV algorithms) to get the normalized similarity between frames and plot them in green lines. By observing the R,G,B lines in each figure, we can find



(a) Parallel Translation(R=100)



(b) Vertical Translation(R=20)

Fig. 4. Translation similarity(Theoretical V.S. Practical V.S. CV Algorithm)

that lines in each figure share a similar trend in descending. $Sim_{\perp}$ decreases more rapidly than $Sim_{\|}$, which is consistent with our theoretical analysis.

In real circumstances, all the situations can be seen as a combination of rotation and translation. Fig. 5 illustrates the comparison between FoV based similarity and frame differencing based similarity in different situations. In each subfigure, the color of red indicates that the corresponding two FoVs(or frames) are highly similar while the color of blue indicates that they are less similar to each other.

Case 1: Rotation

In this experiment, the client user holds the video camera and rotates. The diagonal in the FoV based similarity rectangle in Fig. 5(a) reveals the previous hypothesis. When the user is rotating, the similarity between two FoVs will decrease with the rotation angle θ. When θ surpasses the viewing angle of the smart phone camera, the similarity will stays as 0 since there is no intersection between these two FoVs. It is also applicable when using vision algorithm.

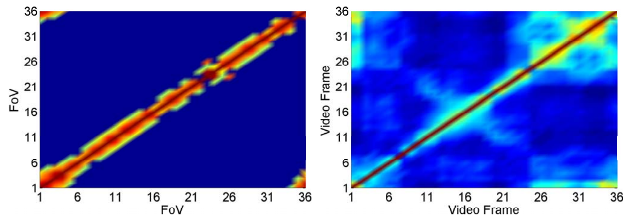
Case 2:Translation

The experiment of translation situation is performed by a user driving down the street and capturing the view in front of car. So we make $R = 100m$. The similarity will decrease gradually when the driving distance d grows. We can discover this pattern by scanning the similarity rectangles horizontally from each subfigure in Fig. 5(b). In this situation, the similarity won't drop to 0 even the translation distance is long.

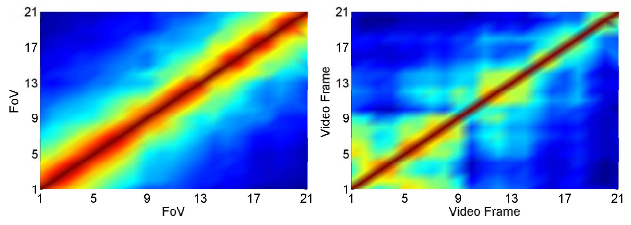
Case 3:Reality

In this experiment, the user captures the video when riding his bike in a residential area. In the middle of the ride, he turns right so the FoV similarity rectangle in Fig. 5(c) is divided into four parts. FoVs in the first half and second half have no intersections so the upper-left and lower-right parts are in blue. This pattern can also be found in the vision based similarity rectangle on the right. We can see that the blue cross in the rectangle reveals the turning event. Though the upper-left and lower-right of the rectangle shows that there are still similar pixels, it represents nothing since FoV already tells us that the field of view has completely changed.

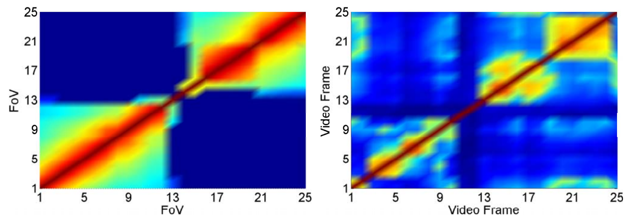
2) Video Segmentation:



(a) Rotation Situation



(b) Traslation Situation



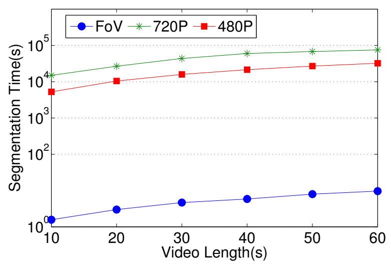
(c) Reality

Fig. 5. Correlation between the two similarity measurement

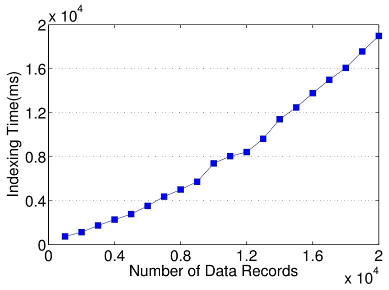
Video segmentation will be executed on the smart phones while the users are capturing videos. To evaluate the performance of our algorithm, we implement the off-line editions of FoV based and CV based segmentation algorithms.

Fig. 6(a) plots the efficiency of different video segmentation approaches. Apparently, it is more time-consuming when the videos are in higher resolution. As FoVs are resolution-independent, it is at least three orders of magnitude faster to perform an FoV based segmentation on a video of same length.

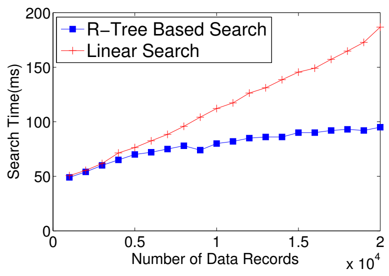
To evaluate the indexing and retrieving performance, we



(a) Efficiency of segmentation approaches



(b) Setup Time



(c) Retrieval Efficiency

Fig. 6. Segmentation, Indexing and Retrieval

randomly simulate citywide representative FoVs and perform insertion and search operations on the tree structure.

Fig. 6(b) plots the time performance of setting up the index. It takes no more than 20 seconds to insert 20,000 records on an off-the-shelf laptop. In other words, on average, it just cost milli-seconds to insert a new incoming representative FoV.

Then we compare the searching performance of our index structure with the naive linear search. The definition of FoV has greatly reduced the data size of the video descriptor and computation cost on video segmentation, and the R-Tree based indexing structure will further reduce the searching cost. As Fig. 6(c) illustrates, when the number of data records is small, the cost of performing a linear search is close to that of performing an R-Tree based search. However, when the size of data grows bigger, the advantage of R-Tree gradually emerges.

VII. Discussion

Radius of View and Segmentation Threshold

Radius of view R and the segmentation threshold $thresh$ can affect the sensitivity of FoV similarity measurement and video segmentation. Intuitively speaking, the similarity would decrease slower when R gets bigger and when threshold gets bigger, the segmentation of video would be denser.

Both arguments will affect the performance of video retrieval. In our system, the assignment of R and $thresh$ is determined by our empirical observation. However, there are possibilities to adaptively assign the values of R and d. For example, Google Maps can help us do the site survey. By analyzing the visual features on the map, radius of view and segmentation threshold could be estimated.

Video Utility and Incentive Mechanism

Our system is a transformation of Mobile Crowdsourced Sensing(MCS) application. The utility of a video according to a query can take the utility of angular coverage U_a and temporal coverage U_t into account. In this case, for a query Q, the global utility can be defined as $360° \times (t_e - t_s)$ and the utility of a certain video is the size of the sub-rectangle area which depend on the U_a and U_t of the representative FoV. Since the utility rectangles of different videos may have intersections, this utility function is non-negative monotone submodular. The utility of a FoV set $U(S)$ is the size of the polygon, which is the union of all the utility rectangles of the FoVs in S.

With our definition of utility, the interaction between client user and cloud server can be considered as a *zero arrival-departure interval* case, and we can set up an incentive mechanism when the inquirer has a reserved budget.

VIII. Related work

Crowd-Sourced Video Systems

Crowd-sourced video retrieval needs the collaboration between server and client users(smartphone subscribers). The existing systems entrust the client users upload the videos and make the server do the video collection, management and retrieval. [12] proposes a mobile video streaming framework, which provides efficient video streaming services for each mobile user. [13] presents an effective car video retrieval system. The vehicles transfer the videos captured by driving recorder up to the mobile video cloud and the server adopts SIFT feature to search the probable video segments of traffic accidents. [14] implements the scalable system GigaSight,which is a hybrid cloud architecture, to manage first-person video from mobile devices. To cope with privacy issues, the author propose denaturing the privacy content using computer vision algorithms. Similarly, in [15] videos captured by smart glasses are uploaded to YouTube and all the corresponding information(including URL) are maintained in a SQL database. [16] proposes a crowdsourcing based video traffic surveillance framework. The system exchanges video clips with smartphone users to obtain an intelligent traffic surveillance. However, all these systems never take the rich geo-information of the crowd-sourced videos into consideration. Besides, we can never guarantee the network environment around the users. Under this uncertainty, uploading raw video data become unrealistic for smartphones.

Multimedia Descriptors

Retrieval for multimedia has been a research hotspot since 1990s. Generally speaking, descriptors used to represent the content of the multimedia data fall into two categories, *global features* and *local features*. Global features,*e.g.* color Gist [17], HLAC [18], focus on the overall distribution of an image and they can satisfy some low-level content-based retrieval in a short period of time. On the other hand, SIFT(Scale Invariant

Feature Transform) [19] and its variants [20] [21]are among the most frequently used local features. The extraction of feature points enables the robust and accurate description of image and video contents. However, it also brings high time space complexity. Accelerating SIFT descriptor has been being a research issue up to now. Resource-limited smart phones cannot afford the heavy burden of computation.

FoV Based Video Systems

Multimedia with geo-information has become a research issue in 2003 [22].FoV helps us manage the multimedia data better. [8] modeled the viewable scene in a geospatial video, which is the prototype of FoV. In [8], viewable scenes are estimated as rectangles. Later, FoV is remodeled as a vector in [23] to improve the search accuracy. Considering the distance between FoV and the query place, [24] gives a rank to the FoVs that have intersections with the query range. The research work above have propose the theoretical system of FoVs. However, all of them think of FoV as a static area, omitting that the FoV of a mobile video could change along with the user's movement. Moreover, videos are time-sensitive to the query as well as the corresponding FoVs. All of the proposed search schemes cannot be applicable in mobile, time-sensitive situations.

IX. Conclusion

In this paper, we propose a rational similarity measurement over FoVs, which shows high efficiency and good compliance with the visual descriptors. Based on our measurement, real-time video segmentation and segment abstraction algorithms are implemented on smart phones. Moreover, a dynamic time-sensitive FoV indexing structure is maintained on the cloud side. Inquirers can activates a search request by a spatio-temporal interval and the server will return a rank based result based on the quality of each mobile video segment.

X. Acknowledgement

This work is supported by NSF China Major Program 61190110 and National High Technology Research and Development Program of China (863 Program) under Grants No.2015AA01A201. Kebin Liu is the contact author. We thank all the reviewers for their valuable comments and helpful suggestions.

References

[1] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 6, pp. 797–819, 2011.

[2] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004. [Online]. Available: http://dx.doi.org/10.1023/B%3AVISI.0000013087.49260.fb

[3] B. A. Fisher and D. R. Fisher, *Techniques of crime scene investigation*. CRC Press, 2012.

[4] D. Gutchess, M. Trajkovics, E. Cohen-Solal, D. Lyons, and A. K. Jain, "A background model initialization algorithm for video surveillance," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 733–740.

[5] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 139–152.

[6] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 389–400.

[7] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Using crowd-sourced viewing statistics to save energy in wireless video streaming," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 377–388.

[8] S. A. Ay, R. Zimmermann, and S. H. Kim, "Viewable scene modeling for geospatial video search," in *Proceedings of the 16th ACM International Conference on Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 309–318. [Online]. Available: http://doi.acm.org/10.1145/1459359.1459401

[9] S. Arslan Ay, L. Zhang, S. H. Kim, M. He, and R. Zimmermann, "Grvs: a georeferenced video search engine," in *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 2009, pp. 977–978.

[10] H. Ma, S. Arslan Ay, R. Zimmermann, and S. Kim, "Large-scale geo-tagged video indexing and queries," *GeoInformatica*, pp. 1–27, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10707-013-0199-6

[11] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '84. New York, NY, USA: ACM, 1984, pp. 47–57. [Online]. Available: http://doi.acm.org/10.1145/602259.602266

[12] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. Leung, "Ames-cloud: a framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 811–820, 2013.

[13] K.-H. Lee, J.-N. Hwang, J.-H. Yoo, and K.-H. Choi, "Effective car video retrieval using feature matching in a mobile video cloud," in *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, Oct 2012, pp. 1–6.

[14] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 139–152. [Online]. Available: http://doi.acm.org/10.1145/2462456.2464440

[15] Z. Chen, W. Hu, K. Ha, J. Harkes, B. Gilbert, J. Hong, A. Smailagic, D. Siewiorek, and M. Satyanarayanan, "Quiltview: a crowd-sourced video response system," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, p. 13.

[16] H. Wen, Q. Li, Q. Han, S. Ge, and L. Sun, "Poster: Crowdsourcing for video traffic surveillance," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14. New York, NY, USA: ACM, 2014, pp. 384–384. [Online]. Available: http://doi.acm.org/10.1145/2594368.2601460

[17] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001. [Online]. Available: http://dx.doi.org/10.1023/A%3A1011139631724

[18] N. Otsu and T. Kurita, "A new scheme for practical flexible and intelligent vision systems." in *MVA*, 1988, pp. 431–435.

[19] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150–1157 vol.2.

[20] G. Zhao, L. Chen, G. Chen, and J. Yuan, "Kpb-sift: a compact local feature descriptor," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 1175–1178.

[21] A. Abdel-Hakim and A. Farag, "Csift: A sift descriptor with color invariant characteristics," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 1978–1983.

[22] K. Rodden and K. R. Wood, "How do people manage their digital photographs?" in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2003, pp. 409–416.

[23] S. H. Kim, S. A. Ay, B. Yu, and R. Zimmermann, "Vector model in support of versatile georeferenced video search," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. ACM, 2010, pp. 235–246.

[24] S. Arslan Ay, R. Zimmermann, and S. Kim, "Relevance ranking in georeferenced video search," *Multimedia Systems*, vol. 16, no. 2, pp. 105–125, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00530-009-0177-x