# Listen, Look, and Gotcha: Instant Video Search with Mobile Phones by Layered Audio-Video Indexing*

Wu Liu [1,3], Tao Mei [2], Yongdong Zhang [1], Jintao Li [1], Shipeng Li [2]
[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China
[2] Microsoft Research Asia, Beijing 100080, P. R. China
[3] University of Chinese Academy of Sciences, Beijing 100049, P. R. China
liuwu@live.cn; {tmei, spli}@microsoft.com; {zhyd, jtli}@ict.ac.cn

## ABSTRACT

Mobile video is quickly becoming a mass consumer phenomenon. More and more people are using their smartphones to search and browse video content while on the move. In this paper, we have developed an innovative instant mobile video search system through which users can discover videos by simply pointing their phones at a screen to capture a very few seconds of what they are watching. The system is able to index large-scale video data using a new layered audio-video indexing approach in the cloud, as well as extract light-weight joint audio-video signatures in real time and perform progressive search on mobile devices. Unlike most existing mobile video search applications that simply send the original video query to the cloud, the proposed mobile system is one of the first attempts at instant and progressive video search leveraging the light-weight computing capacity of mobile devices. The system is characterized by four unique properties: 1) a joint audio-video signature to deal with the large aural and visual variances associated with the query video captured by the mobile phone, 2) layered audio-video indexing to holistically exploit the complementary nature of audio and video signals, 3) light-weight fingerprinting to comply with mobile processing capacity, and 4) a progressive query process to significantly reduce computational costs and improve the user experience—the search process can stop anytime once a confident result is achieved. We have collected 1,400 query videos captured by 25 mobile users from a dataset of 600 hours of video. The experiments show that our system outperforms state-of-the-art methods by achieving 90.79% precision when the query video is less than 10 seconds and 70.07% even when the query video is less than 5 seconds.

## Categories and Subject Descriptors

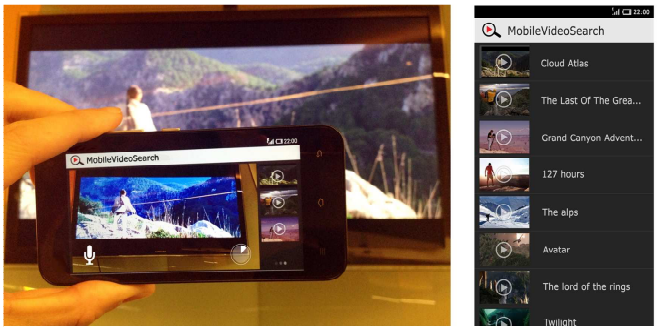
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Video*

## General Terms

Algorithms, Performance, Design, Experimentation.

## Keywords

Mobile Video Search, Audio-Video Signatures, Layered Audio-Video Indexing, Progressive Query Process, Instant Search.




(a) screenshot of mobile video search (b) instant search results

**Figure 1: A snapshot of our proposed mobile video search system. (a) A user simply points his or her phone at a screen to capture a few seconds of what they are watching. While recording, the user can see instant search results on the right. (b) The search process can stop anytime once a confident search result is achieved. Thus, the user does not need to wait for a fixed time lag. The proposed system is characterized by its unique features such as layered audio-video indexing, as well as instant and progressive search.**

## 1. INTRODUCTION

Mobile is reshaping the way video content is generated, searched, and consumed on today's Internet. The proliferation of increasingly capable mobile devices opens up exciting possibilities for mobile video applications. As a result, mobile video is quickly becoming a mass consumer phenomenon. Among many mobile video applications, mobile video search—searching similar or duplicate videos to identify a phone-captured query video—is emerging and becoming pervasive as: 1) while on the go, users always prefer mobile devices as their principal video search and browsing

* This work was performed at Microsoft Research Asia.


*ACM MM* '13, October 21-25, Barcelona, Catalunya, Spain.

http://dx.doi.org/10.1145/2502081.2502084.

tools [4], and 2) the advanced built-in cameras have made video search very natural—mobile users can now discover videos by simply pointing their phones at a screen to capture a few seconds of what they are watching, as shown in Figure 1(a).

While extensive research has been made on mobile visual search (e.g., landmark recognition [9], product search [5], augmented reality [6], and so on), mobile video search is still in the early stages and remains a challenging research problem. Unlike traditional desktop video search, video search on mobile devices faces the following unique challenges: 1) *large aural-visual variance of query video*—due to the complex capture conditions, the query clip is naturally noisy with varying aural and visual qualities. Therefore, designing robust signatures that can deal with such significant variance is crucial for mobile video search. 2) *stringent memory and computation constraints*—as the CPU and memory of mobile devices are still not comparable with desktop computers, signatures with large memory costs or heavy computation are not suitable for mobile clients. 3) *network bandwidth limitations*—with low and unreliable bandwidth, signatures are expected to be as compact as possible to reduce network latency, and 4) *instant search experience*—because mobile users care more about their experience than in desktop search, the search process is expected to be instant and progressive; in other words, the captured query clip needed for search should be as short as possible, while the search results should be returned instantly or even progressively.

Many existing mobile applications have tried to provide video search functions. For example, IntoNow enables mobile phone users to record a 12-second audio clip as the query [7]. Although no algorithm details are available, it is likely that the search is accomplished by audio fingerprinting technology. However, as it heavily depends on audio information, it may not work well if the recording conditions are noisy or the recorded query video is silent. On the other hand, VideoSurf provides a video search service based on the visual information extracted from a 10-second recorded video clip [24]. Nevertheless, it needs users to choose the discriminative visual content for search. It is also worth noting that these applications fully rely on the computing power of the cloud by simply sending the original query video to a server, while neglecting the increasing computing capacity of mobile clients. As a result, users need to record a video clip of a fixed duration (e.g., 10 or 12 seconds). This paper investigates if we can leverage the computing capacities of advanced smartphones for extracting robust and compact audio-video signatures on the mobile client. In this way, instant and progressive video search can be achieved, where the duration of the query video is less than those in any existing systems and the search process can stop anytime once a confident result is achieved.

Although mobile visual search has attracted extensive attention in the research community [2, 4, 5, 8], few efforts have explored mobile video search. The most related research to video search on mobile devices is near-duplicate video search, which can be classified into three categories according to the type of data modality they rely on: audio-based [10, 25], video-based [18, 20], and fusion-based methods [21]. However, most existing approaches to duplicate video search predominantly focus on desktop scenarios where the query video is usually a subset of the original video without significant distortions. Moreover, the computational costs and compactness of descriptors are often neglected in those systems. Therefore, conventional approaches to duplicate video search have not taken the aforementioned mobile challenges into account, and thus are not suitable for mobile video search.

Motivated by the above observations, and aiming to address the aforementioned mobile challenges, we have developed an innovative instant mobile video search system through which users can discover videos by simply pointing their phones at a screen to capture a few seconds of what they are watching. In the cloud, the system is able to index large-scale video data using a novel Layered Audio-VidEo (LAVE) indexing scheme; while on the client, the system extracts light-weight joint audio-video signatures in real time and searches in a progressive way. Specifically, the LAVE scheme provides a new way to combine audio-video signatures through joint multi-layered audio-video indexing, which preserves each signature's individual structure in the similarity computation and considers their correlation in the combination stage. The joint audio-video signature is not only computationally cheap for mobile devices, but also mutually reinforces the discriminative power from individual modalities and thus is robust to the large variance (noise and distortion) in the query video. The learned hash function significantly reduces the bits to transfer from mobile to server. The design of the search process via a bipartite graph transformation and matching algorithm makes the video search progressive—the search can stop anytime once a confident result is achieved. The experiments show that our system can achieve 90.77% precision when the query video is less than 10 seconds and 70.07% when the query video is less than 5 seconds. Figure 1 shows a screenshot of the system.

In summary, this paper makes the following contributions:

- We have designed an innovative mobile video system that represents one of the first attempts towards instant and progressive video search, by leveraging the light-weight computing capacity of mobile devices. This is unlike most existing mobile video search systems.
- We propose a new layered audio-video indexing scheme to holistically exploit the complementary nature of audio and video signals for more robust video search.
- We propose a progressive query process to support varying lengths in the query video, where in most cases the length is much shorter than those in any existing mobile applications. This significantly reduces query time and thus improves users' search experience.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed mobile video search system. Section 4 describes the experiments and evaluations, followed by the conclusion in Section 5.

## 2. RELATED WORK

We next review related research on near-duplicate video search and mobile visual search, which are closely related to the design of a mobile video search system.

### 2.1 Near-Duplicate Video Search

Although mobile video search is still in an early research stage, there has been intensive research on general near-duplicate video search. Depending on the query modality,

existing approaches mainly fall into three categories: audio-based, video-based, and fusion-based approaches.

Extensive research has been conducted on using visual signatures to search near-duplicate videos. A widely adopted signature is global features [16, 18], where videos are represented by compact global signatures. For example, Shang *et al.* propose a compact spatiotemporal feature that leverages gray-level intensity distribution with respect to timeline to represent videos [18]. Similarly, Paisitkriangkrai *et al.* combine both spatial and temporal information into ordinal measures to construct compact and invariant global signatures [16]. Although these global representations achieve fast retrieval speeds in a large-scale video dataset, they usually become less effective in handing recorded query videos with serious distortions [28]. Compared with the global features, local descriptors are more distinctive and robust to these video distortions as they explore the local invariance (such as scale and orientation). Therefore, local descriptors are also employed to similar video search. However, due to the computational complexity, the efficiency becomes intractable. There are many approaches improving the speed of local descriptor matching. Bag-of-Words (BoW) is a popular scheme for near-duplicate video search [20]. Furthermore, Wu *et al.* construct a hierarchy structure to speed up the matching process [28]. However, local descriptor-based approaches still cannot be directly adopted on mobile devices without proper optimization due to the limited computing capability and memory of mobile devices.

On the other hand, audio plays an important role for near-duplicate video search [10, 25]. For example, Wang *et al.* propose a simple but effective landmark-based audio fingerprint to conduct similar audio search [25]. Furthermore, inspired by BoW, a bag of audio words (BoA) representation is proposed to characterize audio features for similar video search [10]. Compared with visual features, audio features are robust, computationally efficient, and compact, and therefore suitable to employ in mobile video search.

Recently, joint audio-visual near-duplicate video search has attracted attention as audio and video can complement each other. In particular, the TRECVID community has tried to explore the benefit of the audio-visual combination for large-scale video copy detection. The key problem of feature combination is the identification of the correlation between audio and video features. Early and late fusion strategies are widely used fusion methods [21]. However, both strategies have disadvantages—early fusion cannot well preserve the individual structural information of each individual feature while late fusion does not consider the correlation among features [21].

In summary, fusion of audio and visual features gives a neoteric avenue to near-duplicate video search. However, existing early or late fusion methods cannot sufficiently mine the advantage of audio-video signatures. In addition, although the existing near-duplicate video search methods can be used as references, few of them can be directly adapted in mobile video search to deal with unique mobile challenges.

## 2.2 Mobile Visual Search

Mobile visual search has attracted extensive attention due to its huge potential for numerous applications. Most research on this topic has predominantly focused on compact descriptor design on mobile devices. The most popular way to solve this problem is compressing descriptors through the technology of image coding. For example, the Compressed Histogram of Gradients (CHoG) encodes the raw features with 20x reduction by an entropy-based coding method on the mobile client [2]. On the server, the compressed descriptors are approximately decoded. Similarly, Ji *et al.* propose a multiple-channel coding scheme to extract compact visual descriptors [8]. In contrast, He *et al.* [5] and Tseng *et al.* [23] have suggested using the hashing function to compress the descriptors. Compared to the above schemes, the hash bits of the local features do not need to be decoded on the server, which can be directly viewed as a visual word index. In addition, it offers lower transmission costs, and cheaper memory and computation than other methods. However, there are few methods for mobile video search. Although Chen *et al.* design a dynamic query selection algorithm for mobile video retrieval [3], using only one keyframe can hardly handle the large variance of the query video captured in complex conditions. Furthermore, sending raw local features is also time and energy consuming. Wu *et al.* employ a similar video search in their mobile video question-answering system, which mainly address different challenges [27].

In summary, compared to the above methods, our proposed system aims to provide instant and progressive mobile video search. We have designed the search scheme to progressively transmit compact hash bits, which are derived from the joint and light-weight audio-video signature, to the cloud. The proposed LAVE indexing technique exploits the advantage of the audio-video signature for robust video search. Moreover, to improve users' search experience, a progressive query process via a bipartite graph based transformation and matching method has been developed.

# 3. INSTANT MOBILE VIDEO SEARCH

## 3.1 Overview

Figure 2 shows the architecture of the proposed mobile video search system, including the offline and online stages. In the offline stage, we first extract the audio-video descriptors for each video from a large-scale source video dataset. Then, these massive descriptors are indexed by a novel LAVE indexing method. The online query stage consists of the following steps: 1) Real time extraction of light-weight audio-video descriptors on mobile devices, while users are capturing query video clips. The signatures (including visual hash bits and audio fingerprint) are sent to the server at an interval of one second. 2) Once the server receives the one-second signature, the search for similar video keyframes is conducted through the LAVE index. 3) A fast and effective geometric verification-based visual re-ranking step is used to refine the search results. 4) A progressive query process via bipartite graph transformation and matching is performed to make the video search progressive. If there are no change in the instant search results for three consecutive seconds (i.e., a confident search result is regarded to have been achieved), then the search process can stop and the results will be returned to the users. In the next section, we describe each component in detail.

## 3.2 Joint Audio-Video Descriptor

In contrast to other media, the main advantage of video is that it contains both abundant audio and visual information. As the complementary nature of the audio and video signals, the joint audio-video descriptors are more robust

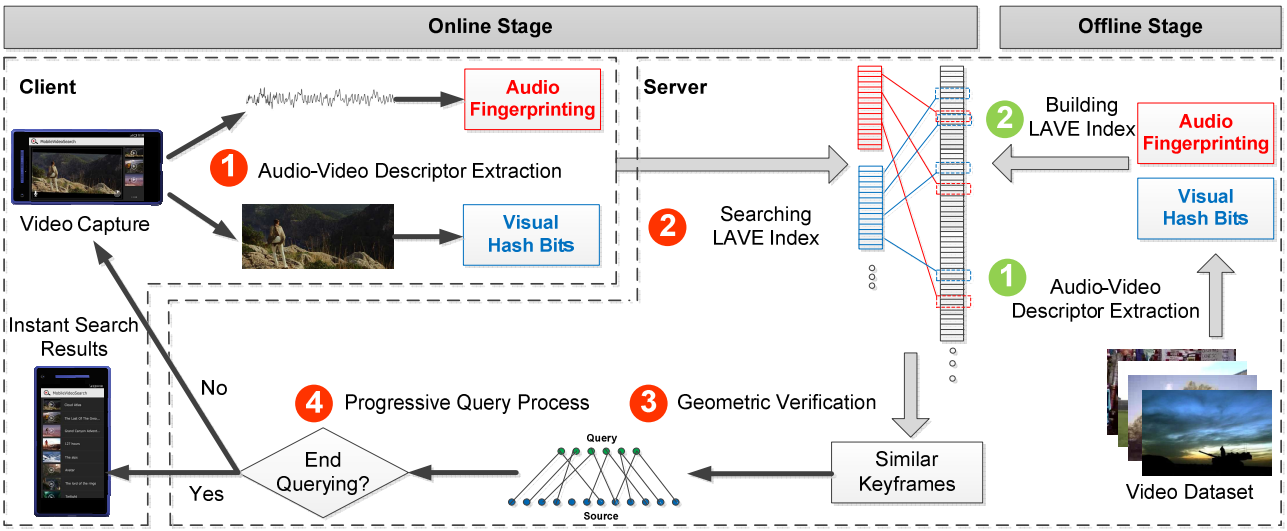



**Figure 2: The overview of our proposed mobile video search system.**

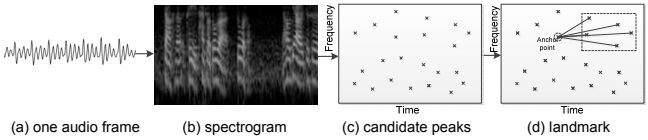



**Figure 3: The extraction of landmark-based audio fingerprinting.**

to the large variance of query videos, especially for complex mobile video capturing conditions (e.g., silent video or blurred video of low visual quality). Selecting effective audio-video descriptors is very important for a mobile video search system. There are three main principles for joint descriptor selection: 1) robust to the variance of the recorded query videos, 2) cheap to compute on mobile devices, and 3) easy to index for instant search. Consequently, we choose the Landmark-Based Audio Fingerprinting (LBAF)[25] and Speeded-Up Robust Features (SURF) [1] as follows.

### 3.2.1 *Audio Fingerprinting*

Among various audio features, LBAF is widely used in many near-duplicate video search methods. Its fast computation, efficient memory and invariant translation are also very suitable for mobile video search. In our system, we extract the LBAF as shown in Figure 3. First, the audio information is segmented into short and partly overlapping frames of length $f_{m_l}$ and stride $f_{m_d}$. Second, for each frame, its spectrogram is calculated. Third, the candidate peaks are settled on the spectrogram of the frame according to three criteria: higher energy content than all its neighbors, higher amplitude than its neighbors, and a density criterion [25]. Fourth, anchor points are chosen from the peaks and each anchor has a fixed target zone. Each anchor point is sequentially paired with the candidate peak in its target. The pairs are called landmarks. Each landmark is represented as $l_i = \{t_i^a, f_i^a, \Delta t_i^a, \Delta f_i^a\}$, where $t_i^a$and $f_i^a$ are the time offset and the frequency of the anchor point, and $\Delta t_i^a$ and $\Delta f_i^a$ are the time and frequency differences between the anchor point and the paired point in the target zone. Finally, we compress the fingerprint into $l_i = \{h_k^a, t_i^a\}$ where $h_k^a$ is the hash value of the $f_i^a, \Delta t_i^a$ and $\Delta f_i^a$. Different $l_i$ may have the same $h_k^a$.

In our experiments, we set $f_{m_l} = 256$ ms and $f_{m_d} = 32$ ms, and limit hash bits $h_k^a$ to less than 25 bits. As there are also 15 bits for $t_i^a$, the length of $l_i$ is 40 bits. For a one-second audio clip, we choose 100 landmarks in total. Hence, we only have to transmit 0.5 KB per second for audio fingerprinting.

### 3.2.2 *Visual Hash Bits*

According to our collected real-word mobile video query dataset (to be described in section 4.1), the recorded videos have great distortions, such as blurs, color change, reflection, and transformation. So the video descriptor in our system should be not only quickly extracted on the mobile device, but also robust to these distortions. According to previous research on mobile visual search [5, 8, 30], the SURF is a competent feature for achieving both high efficiency and robustness. Furthermore, accelerating SURF extraction methods on mobile devices could also be used [30].

However, directly sending raw SURF features is very time and energy consuming. Inspired by Bag of Hash Bits (BoHB) [5, 23], hashing methods are used to compress the local features to hash bits, such as its light computation and memory requirements. Here, Minimal Loss Hashing [14] or Spectral Hash [26] can be used to learn the hash function $h^v = sign(v^T x - t)$, where $x$ is the SURF descriptor vector, $v$ and $t$ is the learned hash matrix and the threshold scalar, and $h^v$ is the learned visual hash bits. According to [5] and [23], the binary code is limited to 80 bits. Finally, we use eight bits to save the angle value of the SURF descriptor, which will be used for geometric verification in the future. Therefore, for each SURF feature, we will compress it to $v_i = \{h_l^v, r_i^v\}$, which is only 88 bits in length.

On the other hand, because of the different camera resolutions on mobile devices, we will scale the query image to a small picture. There are two main advantages: First, it improves feature extraction speed on the mobile device. Second, it decreases the number of feature points that need to be transmitted. According to our experiments, the scaling has little influence on precision but seriously improves query speed. After the scaling, there are only 75 SURF points in one frame on average. In other words, the system only needs to transmit less than 1 KB of visual features to the server.

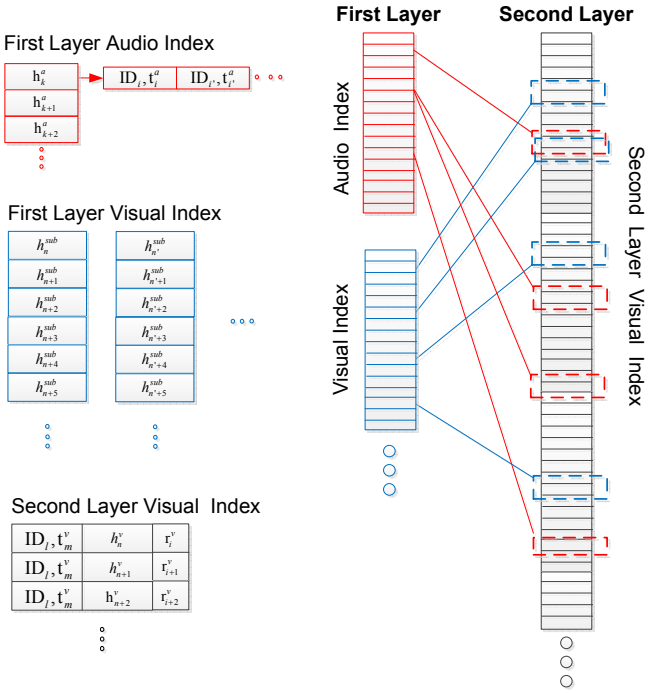



**Figure 4: The illustration of layered audio-video indexing (LAVE).**

In summary, after feature extraction, we obtain 100 audio feature points and 75 visual feature points. Through efficient compression, there are less than 2 KB of audio-visual signatures per second to be transmitted by the network.

## 3.3 LAVE Indexing

Even though the computing of the Hamming distance between binary audio and visual features is efficient, linear search for a big video dataset is still a bottleneck in real-time mobile video search. The most widely used binary feature indexing method is locality sensitive hashing (LSH). However, as it uses random selection, aim to achieve good search quality, it needs a large number of hash bits to build enough tables. As an improvement, Lv *et al.* propose the multi-probe method, which relies on looking through multiple buckets to reduce the number of hash tables [11]. Nevertheless, checking multiple buckets also costs a lot of time. Instead, Muja *et al.* propose the multiple hierarchical clustering trees indexing method [13]. Although the hierarchical decomposition of the search space can greatly reduce search time, it requires multiple trees to avoid the situation in which the closest neighbor to the query point lies across a boundary from the domain explored. As a result, the multiple tree structure increases the memory and query cost.

In our system, we propose LAVE indexing. As shown in Figure 4, there are two layers in the LAVE. The first layer is the index entry, containing a multi-index: audio indexing (red) and visual indexing (blue). The second layer is the visual hash bits, which are used for accurate feature matching and combination. There are two advantages to these structures: 1) effectively employing the hierarchical decomposition strategy to improve the visual points search speed, and 2) holistically exploiting the complementary nature of audio and video signals. The different indexing entries in the first layer preserve the individual structure of audio and video signatures. In the second layer, the combination of audio and video can be weighted by the hamming distance of visual hash bits.

### 3.3.1 Building LAVE Index

In contrast to the visual feature, the audio feature is highly compressed, only 25 bits for each point. Therefore, a linear search of the audio index can be quickly completed. We use the audio index as part of the first layer and each bucket in the audio index of the first layer is associated with the second layer by the video $ID$, audio time offset $t^a$ and keyframe number $t^v$. Through the audio indexing, we can refine the number of visual points to be searched in the second layer, which obviously improves the search speed.

However, if the audio information is changed significantly or missed, it will be difficult to find the closet neighbor in the second layer. The multi-index idea in [15] is used to solve this problem. The hash bits from the second layer visual index are indexed by $m$ different hash tables, which construct the visual index of the first layer. The hash bits $h_n^{sub}$ of the visual index in the first layer are randomly selected from the hash bits in the second layer. For a received visual point, entries that fall close to the query in at least one such substring are considered neighbor candidates. The candidates are then checked for validity using the second layer index. In contrast to [15], we have $m+1$ multi-indexes: visual indexing and audio indexing. Finally, all the results are fused and the top $N$ results are returned. With the help of the audio index, we can greatly reduce the number $m$ for the hash table. In our experiments, even one hash table can still work very well.

### 3.3.2 Searching LAVE Index

The search process in the LAVE indexing is presented as follows. Let $P_a = \{l_1, l_2, \ldots, l_M\}$ be the received audio query points and $P_v = \{v_1, v_2, \ldots, v_L\}$ be the received visual query points. Through the search process, the top $K$ visual points will be returned for each query visual point.

Step 1 For each audio point $l_m$ in $P_a$, the nearest approximate neighbors will be acquired by a linear search in the audio index. Then the matching pairs are assigned to different candidate clusters $C = \{c_1, c_2, \ldots, c_N\}$. Two pairs are assigned to the same cluster if their nearest approximate neighbors come from the same video.

Step 2 All the clusters will be reordered by temporal verification. Here we define the temporal distance $\Delta t$ to denote the time difference of the two LBAFs in the matching pairs. The histogram of $\Delta t$ is computed for all pairs in $c_n$ and the score of $c_n$ equals $h_n/M$, where $h_n$ is the maximum value of the histogram. This score is also used by the similar computation in Section 3.3. Then the top $K'$ candidate clusters are chosen. All the buckets associated with the top $K'$ candidate clusters in the second layer are regarded as a subset.

Step 3 For each $v_l$ in $P_v$, the $K$ nearest approximate neighbors are obtained as follows:

a. Top $K$ approximate neighbors are determined by linear search in the subset of the second layer.

b. Use the multi-index indexing method to search other top $K$ nearest neighbor points.

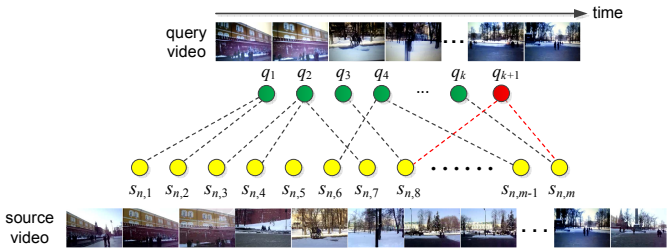



**Figure 5: The examples of bipartite graph between video $v_i$ and queries.**

c. The $2K$ nearest neighbor points are reordered by similar distance. The top $K$ nearest points are selected.

Step 4 Finally, the top $K$ nearest visual points are returned as the search results.

In summary, according to the process, we combine the audio and video in two stages. The first stage is Step 1 – Step 3.a. In this stage, we use the higher compressed audio information as the crude filter and the more discriminative visual information as the fine filter, which obviously improves the final search speed. Furthermore, as the similarity is computed in separate layers, the combination stage also preserves the individual structure of each signature. The second stage is Step 3.b – Step 4. In contrast to the first combination stage, which heavily depends on audio search accuracy, the combination of audio and video can be weighted by the hamming distance of visual hash bits. The two combination stages holistically exploit the complementary nature of the audio and video signals for more robust mobile video search. Finally, as we have $m+1$ multi-index, i.e., $m$ visual indexes and one audio index, the computational complexity of searching the LAVE index mainly depends on the multi-index indexing method used to search the nearest visual neighbor points, which can be found in [15].

### 3.3.3 Geometric Verification

With the top $N$ points, the Hough Transfer method is used to get similar source keyframes of the query. Furthermore, a subsequent geometric verification (GV) considering the spatial consistency of local features is needed to reject the false-positive matches [17]. In order to reduce the time consumption of GV, we follow a fast and effective GV based re-ranking step [29] to find the most similar image. As the method only utilizes the orientation of descriptors, there is no need to transmit the location information of the local features by the network. First, the method hypothesizes two matched descriptors of duplicate images should have the same orientation difference. So for two duplicate images, the orientation distance $\Delta\theta_d$ between each matched local feature pair is calculated. Then all $\Delta\theta_d$ are quantized into $C$ bins ($C = 10$ according to [29]). Furthermore, the histogram is scanned for a peak and the global orientation difference is set as the peak value. Finally, the geometric verification score is given by the number of the pairs in the peak, which is normalized by the number of total pairs.

## 3.4 Progressive Query Process

In contrast to existing mobile video search systems (i.e., search after achieving all the query data), our system provides a progressive query process. Along with the query advancing, the retrieval results can be dynamically calculated on the server after each second query has arrived. Search can stop anytime once a confident result is achieved. The progressive query process can significantly reduce the query cost and improve users' search experience.

**Algorithm 1** *Progressive Query Process*

**Input:** a new query $q_{k+1}$.
**Output:** top $K$ nearest videos

```
add q_{k+1} to Q
search q_{k+1}, get R_{k+1}
add R_{k+1} to R
for each s_{n,m} in R_{k+1} do
    find the G_i contains s_{n,m}
    add q_{k+1} ↔ s_{n,m} to E_i
end for
call W = VideoSimilarScore(G)
return top K nearest videos
```

**Procedure** *VideoSimilarScore(G)*

```
for each G_i in G do
    if |E_i| is changed then
        calculate the MSM M_i
        if |M_i| > α then
            update W_i = Sim(Q, V_i, W_i^a, W_i^v)
        end if
    end if
end for
return W
```

In our system, the progressive query process is implemented via a bipartite graph transformation and matching algorithm. As shown in Figure 5, for each matched query and source video, we use a bipartite graph $G = \{N, E\}$ to represent the matching. In the bipartite graph, the green node $q_k \in Q$ denotes the received query at time $k$, the yellow node $s_{n,m} \in S$ denotes the $mth$ keyframe in source video $V_n$. Let $R_k$ denote all the returned similar keyframes $s_{n,m}$ of query $q_k$. There will be an edge $e_{k,m} \in E$ if $s_{n,m} \in R_k$. After each second search, we can update the bipartite graph $G_i$ and then the similarity score of the matching can be progressively calculated through $G_i$.

The particulars of the progressive query process are shown in Algorithm 1. If one new query comes, a new node will be added. Then, the edges of the bipartite graph are updated according to the returned result. The red node and lines in Figure 4 show this process. For each bipartite graph, if its number of edges has no change, we will skip it. Otherwise, the similarity score of the matched video will be updated as follows: First of all, the Maximum Size Matching (MSM) $M_i$ of $G_i$ is calculated. If $|M_i| > \alpha$, the similar score $W_i$ will be calculated by

$$\begin{aligned} W_i &= Sim(Q, V_i, W_i^a, W_i^v) \\ &= Sim_a(Q, V_i, W_i^a) + Sim_v(Q, V_i, W_i^v) + Sim_t(Q, V_i), \end{aligned} \tag{1}$$

where $Sim_a(Q, V_i, W_i^a)$ favors the audio content similarity, which can be computed as follows,

$$Sim_a(Q, V_i, W_i^a) = \frac{\sum w_{k,i}^a}{|Q|}, \tag{2}$$

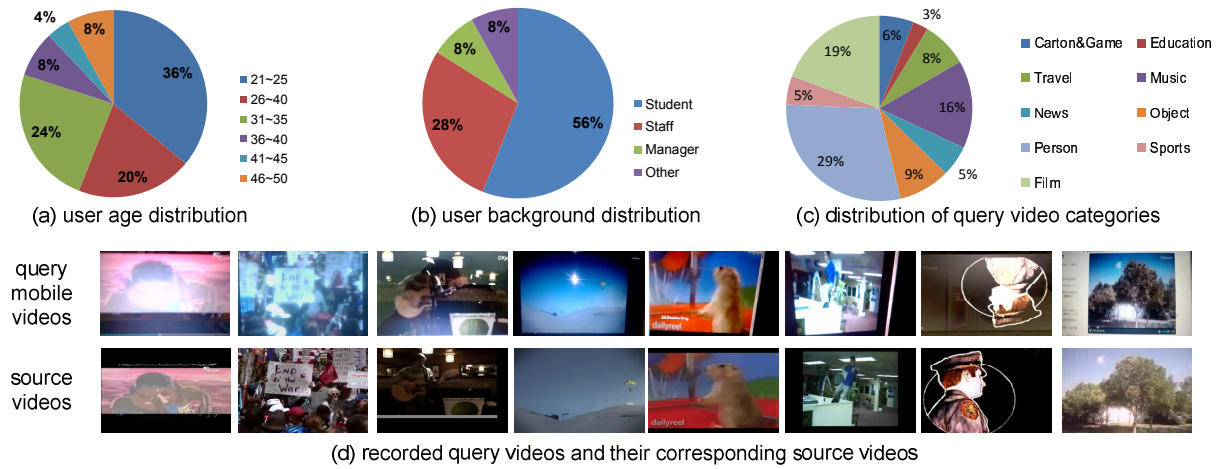



**Figure 6: A dataset of real-world mobile video queries.**

where $w^a_{k,i}$ is the audio similarity between query $q_k$ and video $V_i$, $|Q|$ is the query length. $Sim_v(Q, V_i, W^v_i)$ indicates the visual similarity, given by

$$Sim_v(Q, V_i, W^v_i) = \frac{\sum w^v_{k,i}}{|Q|}, \tag{3}$$

where $w^v_{k,i}$ is the visual similarity between query $q_k$ and video $V_i$. $Sim_t(Q, V_i)$ shows the temporal order similarity. This score guarantees that the matched video should have a similar temporal order. Given MSM $M_i$ of $G_k$, its temporal matching number can be calculated by the Longest Common Subsequence (LCSS) mode [19]. The LCSS is a variation of the edit distance. We use it to denote the number of frame pairs of $M_k$ matched along the temporal order.

$$LCSS(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ LCSS(i-1, j-1) + 1 & e_{i,j} > 0 \\ max\{LCSS(i-1,j), LCSS(i,j-1)\} & e_{i,j} = 0 \end{cases} \tag{4}$$

Thus, $Sim_t(Q, V_i)$ can be obtained by

$$Sim_t(Q, V_i) = \frac{LCSS(|Q|, |V|)}{|Q|}. \tag{5}$$

Finally, after computing all the similarities between $Q$ and $V$, the top $K$ videos will be returned as the search results. The computational complexity of the progressive query process is $\mathcal{O}(|G| \times |N_i| \times |E_i|)$, where $|G|$ is the number of the bipartite graphs, $|N_i|$ and $|E_i|$ is the number of vertexes and edges in each bipartite graph, respectively. However, in actual similarity calculation process, as $|E_i|$ has no change in most bipartite graphs, the time consume is much less.

# 4. EXPERIMENTS

In this section, we first introduce the dataset and queries used for evaluation, and then show the evaluation of the system latency, and the retrieval accuracy. Finally, we conduct user studies to evaluate the usability of our proposed instant mobile video search system.

## 4.1 Real-World Mobile Video Query Dataset

To evaluate the performance of our instant mobile video search system, we use the video dataset in TRACVID 2011, which contains more than 19,200 videos, with a length of 600 total hours [22]. However, in past methods, the query videos are all generated by programs, which are different from real-world mobile video queries recorded by mobile users. Therefore, we asked 25 volunteers to record video clips from the source dataset as queries. In order to better imitate the possible habits of different mobile users, the choice of the volunteers favored diversity: there were eight females and 17 males. Their age distribution is shown in Figure 6(a) and their career distribution is shown in Figure 6(b). In addition, most of the volunteers had at least one year of experience using smart phones. All of them had watched videos on mobile devices, including eight users who frequently had done so. They recorded the videos according to their interests and used different mobile devices (iPhone 4S, HTC z710t, Samsung Galaxy S3, etc.). Finally, we obtained 1,400 query videos with an average length of 30 seconds [1]. Figure 6(c) shows the category distribution of the recorded query videos. Some keyframes of the queries can be seen in Figure 6(d). According to the examples, we found that the recording process produced serious image distortions, such as blurs, color changes, reflections and transformations. The recorded audio signals were also noisy or silent.

## 4.2 Evaluation of System Latency

Latency is very important for instant mobile video search. Basically, our system latency can be broken down into four components: video description extraction latency on the client, video description transmission latency with network, video retrieval latency on the server, and result transmission latency.

- **Video description extraction latency.** We tested the video description extraction time on two types of mobile devices. Device I is equipped with Android 4.1.1 OS, a Qualcomm APQ8064P processor with 1.5 GHz frequency, and a 2GB RAM+16GB ROM. Device II is equipped with Android 4.2.2 OS, a Texas Instruments OMAP 4460 processor with 1.2 GHz frequency, and a 1GB RAM+16GB ROM. Then, we separately extracted the video description from 100 query

[1] The query videos are released in "http://mcg.ict.ac.cn/mcg-mvq.html" as a benchmark for mobile video search in this community.

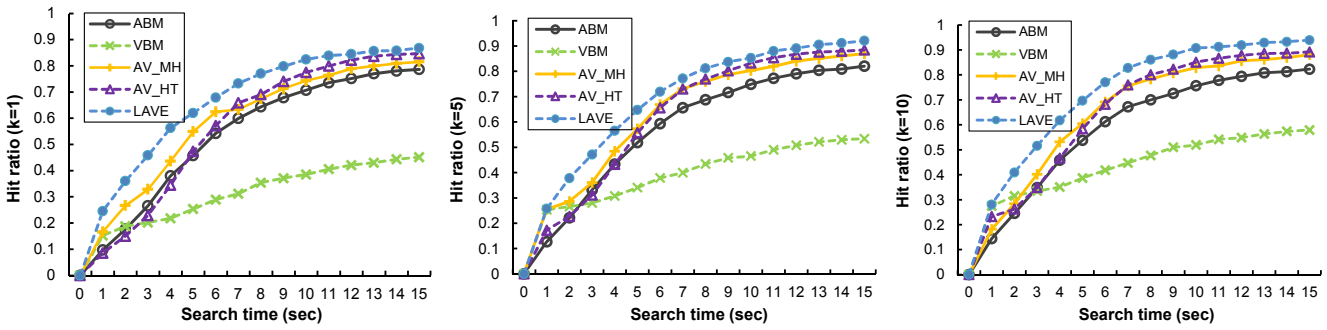



**Figure 7: Comparison of search accuracy on the dataset of real-world mobile video queries (in terms of hit ratio).**

videos on these two mobiles. For each one-second video steam, the average video description extraction time is listed in Table 1. In order to compare, we implemented the visual feature extraction method in [29], which compresses SURF features with a lossless coding method. It cost 398ms on Device I and 417 ms on Device II to extract the visual features, respectively.

- **Video description transmission latency.** With regards to query delivery, we prefer the query bit rate for quantitative computation, as the exact latency in time heavily depends on the type of network used. As introduced in Section 3.2, we hash the audio fingerprinting into 40 bits and totally choose 100 audio fingerprinting points in each second. Additionally, the visual signature is hashed into 88 bits and about 75 visual points are extracted in each second. So we only have to transmit less than 2 KB/s audio-visual signatures by the network. A typical 40 KB/s 3G network only requires 50 ms to transform the data [8].
- **Video retrieval latency.** To test video retrieval latency, we implemented our method on a server that had Intel(R) Xeon(R) CPU with 3.33 GHz frequency, and 48 GB of memory. Then, we retrieved the 1,400 recorded videos on the 600-hour video dataset. The average video retrieval latency for a one-second query includes 119 ms for the search in the LAVE index and 8 ms for the progressive query process.
- **Result transmission latency.** For the list to be complete, we should also add the video search results transmission latency. As we use an image thumbnail and title to represent each video (less than 3KB), we need to transmit 15KB data for the top 5 results to the client. The latency will depend on network setting. It approximately requires 375ms on a typical 40 KB/s 3G network [8].
- **The overall latency.** In our system, the above four components run in parallel. In order to test the whole system latency in the real network setting, we searched 100 different videos with Device I in a real Wi-Fi network and 3G network setting, respectively. The entire system latency was 626 ms with the Wi-Fi network and 781 ms with the 3G network.

## 4.3 Evaluation of Retrieval Accuracy

In order to evaluate our instant mobile video search system, we compared the following five methods on the collected real-world mobile query video dataset.

**Table 1: Time (ms) for extracting video signatures for one-second query clip.**

| Video signatures | Device I | Device II |
|---|---|---|
| Audio fingerprint | 120 | 130 |
| Visual hash bits | 389 | 405 |

(1) **Audio-Based Method (ABM).** The method only uses the audio fingerprinting as a video description and uses linear matching on the server.

(2) **Video-Based Method (VBM).** The method only uses the visual hash bits to describe the video and searches the signatures in the multi-index hash, which is also used in the LAVE.

(3) **AV_MH.** Similar to our method, this method also uses a linear index for audio signature and multi-index hashing for visual signature. The difference is that it combines the audio-video signatures using an average late fusion strategy.

(4) **AV_HT.** As opposed to AV_MH, this method utilizes randomized hierarchical tree indexing for visual signature retrieval. The indexing method is implemented in the Fast Library for Approximate Nearest Neighbors (FLANN) [12].

(5) **LAVE.** Our proposed mobile search approach.

The previous two methods are implemented as baselines for single signature methods. The last three approaches are designed to exploit the complementary nature of the audio and video signals. As the late fusion strategy is widely used by many methods, we use it as the baseline for feature combination methods. To measure the effectiveness of all the approaches, we used the hit ratio, which is defined as the number of correctly match the similar video in the top $k$ results to the total search time. Because we will return the top 10 similar videos to the user, we separately computed the hit ratio when $k = 1$, $k = 5$, and $k = 10$.

### 4.3.1 *Results on the Entire Query Videos*

We first tested the five methods with all 1,400 query videos on the 600-hour video dataset. The hit ratio results of the different methods can be seen in Figure 7. From the results, we found that our method achieved the highest hit ratio among the five methods. Compared to the ABM and VBM, our methods can obviously improve the retrieval accuracy,

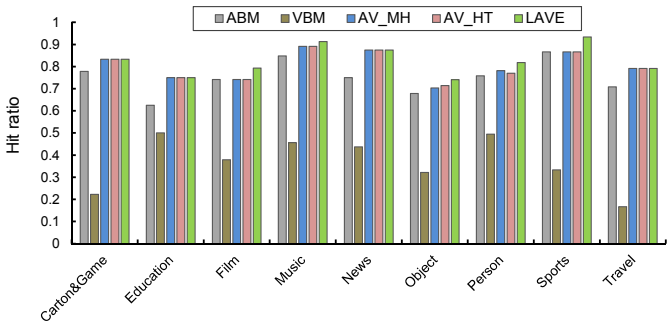



**Figure 8: Comparison of search accuracy on different query categories (in terms of hit ratio).**

which demonstrates that the complementary nature of the audio and video signals reinforce the discriminative power of individual modalities. As a result, our method is robust to the large variance in the recorded query videos. Among the methods combining the audio and video signatures, our method also achieves the highest accuracy. The reason is that the average late fusion strategy used by AV_MH and AV_HT does not consider the correlation among features in the combination stage. In contrast to late fusion, LAVE employs a multi-layer structure to indicate the similarity of audio features by the distance between the visual feature points. In the combination stage, the correlation among features is uniformly measured by the visual distance. So the combination stage of the LAVE scheme effectively chooses the more discriminative features for different queries.

As shown in the Figure 7, we also observe that the performance gain of retrieval accuracy tends to be large for the first ten seconds, while after that, the accuracy grows slowly. We can therefore choose 10 seconds as the longest query time. Our system can achieve 86.08% and 90.77% precision at eight and ten seconds, respectively.

### 4.3.2 *Results on Different Query Categories*

In order to further evaluate the effectiveness of our approach, we also tested the retrieval accuracy on nine different video categories. The query length was set to 10 seconds and the hit ratio was computed from the top five similar videos. As shown in Figure 8, the performance improvements of LAVE were consistent and stable, i.e., most query categories improved compared to the ABM and VBM. Compared to AV_MH and AV_HT, our method also achieved better results for some categories. For example, for the query category "sports," the AV_MH and AV_HT methods got the same hit ratio as the ABM. In other words, the late fusion provided no benefit to the final results. On the contrary, the hit ratio of LAVE was 7% higher than the baseline. The reason is that some sport videos have very similar audio signals. Although the visual signals can discriminate these videos, the average fusion of the audio and video similarity score decreases the information. On the other hand, LAVE indexing methods can easily discriminate these videos through the high weight on the video similarity score. The same situation also occurred in other categories (e.g., "film" and "person").

## 4.4 Subjective Evaluation of Usability

We randomly invited 12 users from the volunteers to use our proposed mobile video search system (LAVES). The subjects included three female and nine male company staff and college students, with ages ranging from 22 to 36. Other information about these subjects can be found in Section 4.1. As they have already participated in the recording task, after only three minutes of orientation and demonstration, all of them understood how to use the system very well. From the interview, we found that 10 subjects thought the instant video search process is very cool when they first saw it. It is worth noting that when they learned our system did not restrict users from recording either audio or video, all the subjects thought it was very convenient and said that they will try to use the system based only on audio or video manually. This indicates that the switch among single-modality-based and multi-modality-based solutions are preferred by users. After learning the system well, the subjects were asked to use the application to accomplish the following tasks.

**Table 2: A summary of the user study comparing three mobile video search applications: (a) IntoNow [7], (b) VideoSurf [24], and (c) LAVES (our method). 1∼5 indicate the worst to the best level.**

| ID | Question | (a) | (b) | (c) |
|---|---|---|---|---|
| 1 | Attractiveness | 3.4 | 3.8 | 4.2 |
| 2 | Naturalness | 3.8 | 3.6 | 4.3 |
| 3 | Practicality | 3.3 | 3.2 | 4.1 |
| 4 | Easy to use | 4.3 | 4.2 | 4.7 |
| 5 | Efficiency | 3.1 | 3 | 4.3 |
| 6 | Effectiveness | 2.8 | 3 | 4.3 |
| 7 | Preference | 3 | 3.2 | 4.2 |
| 8 | Effectiveness of progressive search | — | — | 4.4 |

- Task 1. Each subject selected 5∼10 videos and tried to search for them using LAVES. As our system has not been released to the public yet, the subjects were unable to conduct the search outside the lab. Thus, the query videos were chosen from our video dataset described in Section 4.1. Also, the subjects could search for the video based on audio, video, or both.
- Task 2. In this task, we compared LAVES with popular mobile video search applications (i.e., IntoNow and VideoSurf). After learning how to use these applications, all subjects tried to use the three applications freely by themselves for 20 minutes. Then, a questionnaire was given to each subject to evaluate the usability, user friendliness, and user experience.

In Task 1, the subjects searched 120 videos in total, among which only 16 tasks failed to find similar videos from the top ten similar videos. In addition, all users thought the progressive search process reduced their query time and improved their search experience. According to our records, the average query time for all users was 8.5 seconds. For Task 2, the quantitative evaluation of user satisfaction scores with these mobile video search applications is listed in Table 2. This indicates the advantages of LAVES over the other two mobile video search applications. All the users thought our application was attractive and easy to use with a friendly user interface, and especially liked the lack of restrictions on the use of audio or video as a query input. 91.67% subjects were satisfied with the progressive search process and thought it was natural. They gave a 4.3/5.0 for its Effectiveness. Most of the subjects gave a positive response when

they were asked whether they would install this application and recommend it to their friends. Moreover, the subjects provided comments, such as "add more fashionable videos," "add sharing search results function," and so on.

# 5. CONCLUSIONS

In this paper, we have investigated the possibility of instant video search on mobile devices, where a very short phone-captured video clip is used as the query to identify the captured video from a large-scale video database. We achieved this goal by designing an innovative mobile video search system that leverages the computing power of mobile clients to directly process video query mobile devices. Specifically, we process the video query to extract joint audio-video descriptors for robust video search (superior to a search via a single modality), and compose a compact signature with a new layered audio-video hashing algorithm. We design a novel progressive query process to instantly return search results to mobile users. As a result, the user search experience has been significantly improved with our proposed system due to the state-of-the-art search accuracy and the very short video query.

In the future, we will invite more subjects for field study evaluations to further evaluate the robustness of our system against video queries with large visual and aural distortions. Our future investigations will also include: 1) we will optimize the system in terms of more efficient descriptors, e.g., the speedup SURF optimized on specific mobile platform [30], and indexing schemes; 2) in addition to only search results, we are aiming to integrate social functions (such as recommendation of celebrities, TV programs, and information related to the captured videos) for social applications; 3) we are also aiming to collect real video data in a particular domain (e.g., TV program or movie) in the cloud.

# 6. ACKNOWLEDGEMENT

This work is partially supported by the National Key Technology Research and Development Program of China (No. 2012BAH39B02).

# 7. REFERENCES

[1] H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[2] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod. Chog: Compressed histogram of gradients a low bit-rate feature descriptor. In *CVPR*, pages 2504–2511, 2009.

[3] D. M. Chen, N.-M. Cheung, S. S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod. Dynamic selection of a feature-rich query frame for mobile video retrieval. In *ICIP*, pages 1017–1020, 2010.

[4] B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. A. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham. Mobile visual search. *IEEE Signal Process. Mag.*, 28(4):61–76, 2011.

[5] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung, and S.-F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *CVPR*, pages 3005–3012, 2012.

[6] J. Huber, J. Steimle, and M. Mühlhäuser. Mobile interaction techniques for interrelated videos. In *CHI Extended Abstracts*, pages 3535–3540, 2010.

[7] IntoNow. http://www.intonow.com/.

[8] R. Ji, L.-Y. Duan, J. Chen, H. Yao, Y. Rui, S.-F. Chang, and W. Gao. Towards low bit rate mobile visual search with multiple-channel coding. In *ACM Multimedia*, pages 573–582, 2011.

[9] H. Liu, T. Mei, J. Luo, H. Li, and S. Li. Finding perfect rendezvous on the go: accurate mobile visual localization and its applications to routing. In *ACM Multimedia*, pages 9–18, 2012.

[10] Y. Liu, W. Zhao, C.-W. Ngo, C. Xu, and H. Lu. Coherent bag-of audio words model for efficient large-scale video copy detection. In *CIVR*, pages 89–96, 2010.

[11] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[12] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application*, pages 331–340, 2009.

[13] M. Muja and D. G. Lowe. Fast matching of binary features. In *Proceedings of International Conference on Computer and Robot Vision*, pages 404–410, 2012.

[14] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011.

[15] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. In *CVPR*, pages 3108–3115, 2012.

[16] S. Paisitkriangkrai, T. Mei, J. Zhang, and X.-S. Hua. Scalable clip-based near-duplicate video detection with ordinal measure. In *CIVR*, pages 121–128, 2010.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[18] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua. Real-time large scale near-duplicate web video retrieval. In *ACM Multimedia*, pages 531–540, 2010.

[19] H. T. Shen, J. Shao, Z. Huang, and X. Zhou. Effective and efficient query processing for video subsequence identification. *IEEE Trans. Knowl. Data Eng.*, 21(3):321–334, 2009.

[20] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[21] C. Snoek, M. Worring, and A. W. M. Smeulders. Early versus late fusion in semantic video analysis. In *ACM Multimedia*, pages 399–402, 2005.

[22] TRECVID 2011. http://www-nlpir.nist.gov/projects/tv2011/.

[23] K.-Y. Tseng, Y.-L. Lin, Y.-H. Chen, and W. H. Hsu. Sketch-based image retrieval on mobile devices using compact hash bits. In *ACM Multimedia*, pages 913–916, 2012.

[24] VideoSurf. http://www.videosurf.com/mobile.

[25] A. Wang. An industrial strength audio search algorithm. In *ISMIR*, pages 7–13, 2003.

[26] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[27] G.-L. Wu, Y.-C. Su, T.-H. Chiu, L.-C. Hsieh, and W. H. Hsu. Scalable mobile video question-answering system with locally aggregated descriptors and random projection. In *ACM Multimedia*, pages 647–650, 2011.

[28] X. Wu, C.-W. Ngo, A. G. Hauptmann, and H.-K. Tan. Real-time near-duplicate elimination for web video search with content and context. *IEEE Transactions on Multimedia*, 11(2):196–207, 2009.

[29] Y. Wu, S. Lu, T. Mei, J. Zhang, and S. Li. Local visual words coding for low bit rate mobile visual search. In *ACM Multimedia*, pages 989–992, 2012.

[30] X. Yang and K.-T. T. Cheng. Accelerating surf detector on mobile devices. In *ACM Multimedia*, pages 569–578, 2012.