DYNAMIC SELECTION OF A FEATURE-RICH QUERY FRAME FOR MOBILE VIDEO RETRIEVAL

David Chen[1], *Ngai-Man Cheung*[1], *Sam Tsai*[1], *Vijay Chandrasekhar*[1], *Gabriel Takacs*[1], *Ramakrishna Vedantham*[2], *Radek Grzeszczuk*[2], *Bernd Girod*[1]

[1]Information Systems Laboratory, Stanford, CA
[2]Nokia Research Center, Palo Alto, CA

ABSTRACT

In this paper, we focus on a new application of mobile visual search: snapping a photo with a mobile device of a video playing on a TV screen to automatically retrieve and stream the remainder of the video to the mobile device. When the user takes a photo of the video, the captured query frame may contain too few useful features for good retrieval performance. We design and implement a new algorithm for mobile video retrieval to accurately select a feature-rich frame from a sequence of viewfinder frames in a very short temporal window determined by the user-initiated query event. Fast and accurate selection using efficiently computed Hessian scores is developed for real-time operation on mobile devices. Viewfinder frames captured before the query starts are pre-processed, while the number of viewfinder frames captured afterwards is minimized by a probabilistic optimization process. Evaluated on a large video database of 10 million frames, dynamic query frame selection provides a substantial increase in retrieval accuracy with very low search latency.

Index Terms— Mobile Visual Search, Query Frame Selection, Video Retrieval, Visual Bookmark

1. INTRODUCTION

Many mobile visual search systems have been successfully developed in the past few years for applications like landmark recognition [1] and product cover recognition [2][3]. These systems achieve robust image-based recognition using local features such as SIFT [4], SURF [5], and CHoG [6]. To efficiently search very large databases, a Vocabulary Tree [7][8] is typically employed to index the local features.

Mobile devices are now frequently used to download and watch videos. Yet, these videos are generally disconnected from the video contents watched on the TV screen at home. This disconnect results mainly from a lack of easy-to-use technology which connects the two sources of content. Products from Slingbox [9] provide this functionality through additional hardware at the TV to transcode and forward the video to a mobile device. Instead, one may prefer a software-based solution, where users can simply download a small application to their mobile devices and use a video retrieval service.

In this paper, we demonstrate a mobile video retrieval system that seamlessly connects the videos played on the TV and mobile device, as depicted in Fig. 1. First, a user snaps a photo with the mobile device of the video currently playing on the TV. A compact query signature is extracted from the photo and transmitted to a recognition server. Then, the server searches a large video database and identifies the database video and frame within that video matching the query signature. Subsequently, the server streams the remainder of the video to the mobile device, starting from the recognized frame. This system also allows users to easily store visual bookmarks on their mobile devices for their favorite scenes in movies and TV shows. Using our system, a user can easily transfer the viewing experience from the TV to the mobile phone.

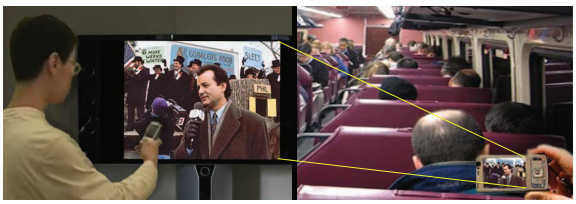
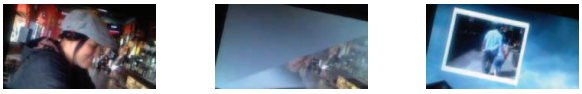


Fig. 1. Mobile video retrieval system where (left) a user takes a photo with a mobile device of a video playing on the TV and (right) the same user resumes watching the same video on the phone, beginning from the recognized video frame. A visual bookmark is also stored on the phone.



(a) 196 features (b) 38 features (c) 82 features

Fig. 2. Frames from the same video clip. Frame (b) trails frame (a) by 0.5 s, and frame (c) trails frame (b) by 0.5 s.

Taking a single photo of a video to perform image-based recognition poses some difficult challenges. There are distortions like camera noise, motion blur, uneven illumination, rotation, cropping, and background clutter. A more serious problem, though, is that when the user snaps a photo of the video, the captured query frame may contain very few useful features for accurate recognition. For example, if the query frame in Fig. 2(b) is selected, that frame contains only 38 SURF features and would be difficult to recognize. Other video retrieval systems [10][11] avoid this problem by using an entire query video (often minutes long) rather than a single query photo to provide good recognition performance. In our low-latency video retrieval system, however, requiring a user to capture a long query video or waiting a long time for the query video to be transmitted and processed would be unacceptable.

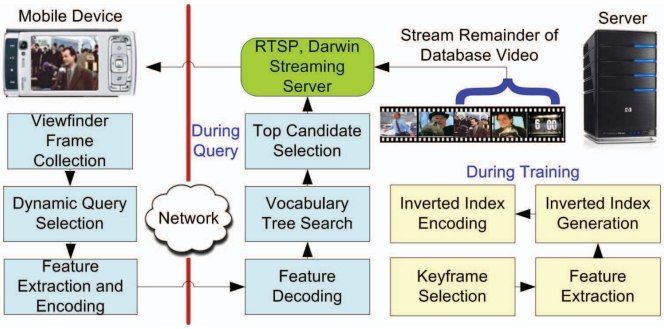
This paper presents a new dynamic query selection algorithm for achieving accurate and low-latency mobile video retrieval. Dynamic selection effectively overcomes the problem of capturing a feature-deficient query frame. In practice, a typical user points the camera at the TV screen for at least a few seconds before and after snapping a photo of the video of interest. By processing the viewfinder frames on the mobile device and looking in a short temporal window

around the instant when the user initiates a query, we can select the viewfinder frame with the highest number of features in that window. Fig. 2(a) and 2(c) show viewfinder frames that occur 0.5 s before and after the frame in 2(b), respectively. Using either 2(a) or 2(c) in place of 2(b) will generally improve recognition. Dynamic selection yields a feature-rich query photo while retaining the advantages of low latencies in query acquisition and transmission.

The remainder of the paper is presented as follows. In Sec. 2, an overview of the video retrieval system is provided. Sec. 3 describes the dynamic query selection algorithm. Our selection techniques are specially designed for fast operation on mobile devices and optimized for minimal query latency. Then, experimental results in Sec. 4 for a large database show the effectiveness of our video retrieval system and the substantial improvements offered by dynamic query frame selection.

2. SYSTEM OVERVIEW

A block diagram for our video retrieval system is shown in Fig. 3(a). On the mobile device, viewfinder frames are first captured. Using the dynamic selection algorithm of Sec. 3, a feature-rich query frame is automatically selected. Then, local features are extracted from the selected query frame and encoded for efficient transmission over the wireless network. Once received on the server, the compressed features are decoded and classified through a Vocabulary Tree. The Vocabulary Tree quickly selects a list of the top 50 matching candidates out of as many as 10 million total frames. Finally, we select the top candidate after applying more rigorous pairwise geometric matching. The recognition result is the identity of database video and frame within that video best matching the query.



(a)

(b)

Fig. 3. (a) Block diagram for our mobile video retrieval system. (b) Our video retrieval client running on a smartphone.

Offline, the database is constructed for fast and accurate recognition. Keyframes are selected from the database videos, to remove temporal redundancy while maintaining good coverage of changing contents. Local features are extracted from each keyframe. The set of all database features is inserted into an inverted index associated with the Vocabulary Tree. An inverted index enables the Vocabulary Tree search to be extremely fast during a query. Finally, inverted index coding [12][13] is applied to significantly reduce memory usage.

Given the recognized database video and frame within that video, the remainder of the video is efficiently streamed from the server to the mobile device. To enable fast download and playback, the Real-Time Streaming Protocol (RTSP) implemented by the Darwin Streaming Server is used to send the recognized database video, starting from the temporal position of the recognized frame. Using RTSP, very low buffering delays (about 1-2 s) are observed on the mobile device before the video starts playing. Fig. 3(b) shows a snapshot of our video retrieval client running on a smartphone. A demo video is available online[1].

3. DYNAMIC QUERY SELECTION

When a query is triggered, our dynamic selection algorithm examines a small temporal window around the user-initiated query event and finds the frame containing the maximal number of features. Suppose the user initiates a query at time t_0, e.g., by pressing a button. Then, the dynamic selection algorithm chooses from the frames falling into the window $[t_0 - \Delta_1, t_0 + \Delta_2]$. In Sec. 3.1, we discuss how to pre-process viewfinder frames falling into the sub-window $[t_0 - \Delta_1, t_0)$. Then, in Sec. 3.2, fast dynamic selection on mobile devices using efficiently calculated Hessian scores will be presented. Finally, Sec. 3.3 describes post-processing of viewfinder frames in the sub-window $[t_0, t_0 + \Delta_2]$ after the query starts and how to minimize Δ_2 while maintaining good selectivity.

3.1. Pre-processing Viewfinder Frames

The viewfinder frames in $[t_0 - \Delta_1, t_0)$ are pre-processed and available when the query starts at t_0, so they do not add to the query latency. We found that a user almost always points the camera at the screen for longer than 1.5 s before t_0, so $\Delta_1 = 1.5$ s is an appropriate choice which provides good selectivity in general. Fig. 4 shows frames from two different 3 s video segments. Numbers below the frames indicate time offsets from t_0. The second row in that figure plots the number of SURF features per frame versus the time offset from t_0. For both video segments, there exists a pre-processed viewfinder frame at a negative time offset that has a significantly higher number of features than the frame at t_0, so dynamic selection would greatly benefit both cases.

3.2. Fast Selection using Hessian Scores

Calculating multi-scale feature interest points at video frame rates on a mobile device is still very expensive. For the purpose of selecting a feature-rich query frame, we prefer to compute another quantity which accurately reflects the number of features and is very efficient to compute. Because SURF interest points are detected as isolated extrema in a Hessian response, we calculate an image-level Hessian score for a frame $I(x, y)$. First, we calculate the following second derivatives for a frame $I(x, y)$:

$$L_{xx}(x, y) = \frac{\partial^2}{\partial x^2}\left[g_0(x, y) * *I(x, y)\right] \quad (1)$$

$$L_{yy}(x, y) = \frac{\partial^2}{\partial y^2}\left[g_0(x, y) * *I(x, y)\right] \quad (2)$$

$$L_{xy}(x, y) = \frac{\partial^2}{\partial x \partial y}\left[g_0(x, y) * *I(x, y)\right] \quad (3)$$

[1] http://www.youtube.com/watch?v=eZWdsP7KeOA

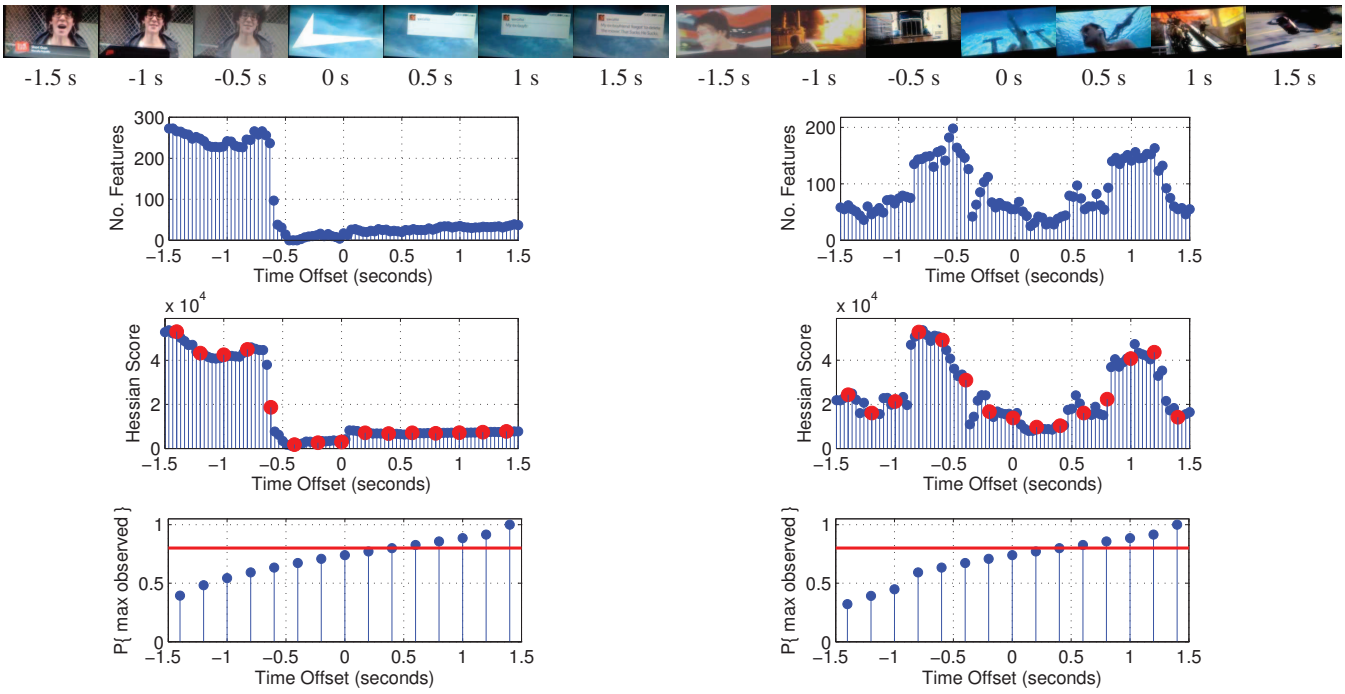


Fig. 4. (1st row) Frames around a user-initiated query event. (2nd row) Number of features. (3rd row) Hessian score, where blue and red dots show sampling at 30 fps and 5 fps. (4th row) Probability that the max Hessian score frame has been observed, with red line showing $t_p = 0.8$.

Here, $g_0(x, y)$ represents a Gaussian at a fine scale where the majority of interest points are detected. Importantly, these filter responses can be approximated very efficiently using an integral image of $I(x, y)$, similar to [5]. Second, we calculate a pointwise Hessian response $H(x, y)$ and then an image-level Hessian score:

$$H(x,y) = L_{xx}(x,y)\,L_{yy}(x,y) - (L_{xy}(x,y))^2 \quad (4)$$

$$\text{Score} = \sum_{x,y} \min\{|H(x,y)|, H_{\max}\} \quad (5)$$

The summation in (5) is over all pixels in the frame and $H_{\max}$ is a saturation value that prevents large pointwise Hessian values from overly skewing the sum.

The 3rd row of Fig. 4 plots the Hessian score of (5) versus time offset from t_0. The blue dots correspond to computing the Hessian score for every frame at 30 fps, while the red dots correspond to Hessian scores computed at a reduced rate of 5 fps. Two key observations can be made to yield critical computational savings on the mobile device. First, there is a very strong correlation between the Hessian score and the actual number of features. The frame with the largest Hessian score in the window has a large number of features. Second, sampling the Hessian scores at 5 fps is sufficient to capture the large changes in Hessian scores in the window.

3.3. Post-processing and Early Termination

Although the pre-processed viewfinder frames in $[t_0 - \Delta_1, t_0)$ do not add to query latency, the post-processed frames in $[t_0, t_0 + \Delta_2]$ do affect the query response delay after the query starts at t_0. Thus, we want to minimize Δ_2, while still guaranteeing with high probability that we select a feature-rich frame. Δ_2 can take a value as large as 1.5 s (same as the value chosen for Δ_1 previously), but on average, we aim to reduce Δ_2 to a much smaller value.

Minimization of Δ_2 is achieved by solving the following optimization problem. By the time a query begins at t_0, the pre-processed frames $-N, -N+1, \cdots, -1$ and their Hessian scores $H_N, H_{-N+1}, \cdots, H_{-1}$ have already been observed. Additionally, after t_0, some post-processed frames $0, 1, \cdots, k$ and their Hessian scores $H_0, H_1, \cdots, H_k$ may also be observed. It is k, the number of frames observed after t_0, that we minimize according to:

$$\min_{0 \le k \le N} \quad k \quad (6)$$

$$\text{s.t.} \quad \Pr\left\{ \max\mathbf{H}_{k+1}^{N} \le h \,|\, \max\mathbf{H}_{-N}^{k} = h \right\} > t_p \quad (7)$$

$$\max\mathbf{H}_{k+1}^{N} = \max\{H_{k+1}, \cdots, H_N\} \quad (8)$$

$$\max\mathbf{H}_{-N}^{k} = \max\{H_{-N}, \cdots, H_k\} \quad (9)$$

The constraint in (7) means that given the previously observed max Hessian score h up to frame k, the probability that future Hessian scores are less than or equal to h is above a threshold t_p. A value of $t_p = 0.8$ has been found to give good early termination results, leading to $\Delta_2 = 0.6$ s on average with little change in recognition accuracy compared to the full search case of $\Delta_2 = 1.5$ s. The probabilities for different values of k, $\max\mathbf{H}_{k+1}^{N}$, and $\max\mathbf{H}_{-N}^{k}$ are learned offline during database construction, by empirically estimating the probabilities using many different training videos.

The 4th row of Fig. 4 shows the probability in (7) versus time offsets from t_0. If we set a threshold $t_p = 0.8$, we can early terminate at $\Delta_2 = 0.4$ s in both cases, much sooner than the max value of $\Delta_2 = 1.5$ s. Using this optimization, we can substantially reduce the query latency caused by searching for the frame with the greatest Hessian score in $[t_0, t_0 + \Delta_2]$, while guaranteeing with high probability that the max-score frame is selected.

4. EXPERIMENTAL RESULTS

We have evaluated our video retrieval system and dynamic query selection algorithm on a database of 2000 YouTube video clips representing 10 million frames. Subsets of these 2000 clips are also used to study changes in recognition accuracy versus database size. Keyframes are uniformly selected by $10\times$ temporal subsampling, yielding 1 million keyframes. SURF features are extracted from these keyframes, and a Vocabulary Tree with 6 levels and 1 million leaf nodes is trained from a randomly selected subset of features. The query data are obtained by recording with a mobile device 50 different database videos playing on a TV screen, yielding 50 query videos containing difficult photometric and geometric distortions. Then, 1224 query frames well separated in time are extracted from the 50 query videos. Ground truth correspondences in the database for these 1224 query frames are precisely established. During a query, a correct match is achieved if the correct database video is reported and the matching frame within that video comes within 10 s of the ground truth frame.

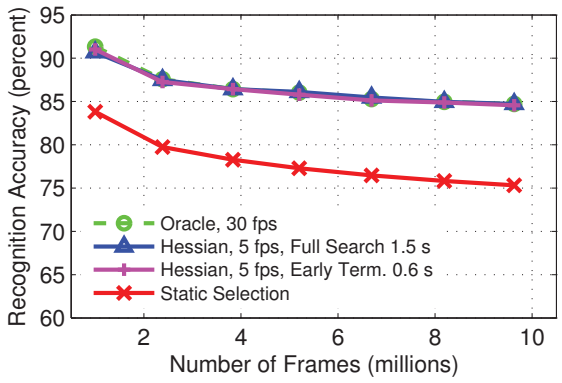


Fig. 5. Recognition accuracy versus number of frames in the video database for four different schemes.

Fig. 5 plots recognition accuracy versus the number of frames in the database for 4 different schemes. First, the "Static Selection" scheme simply uses the viewfinder frame at the exact instant of the user-initiated query, so it is prone to the problem of selecting feature-deficient frames. The three other schemes are variations of the dynamic selection algorithm. Second, an "Oracle" scheme knows the number of features for each viewfinder frame in a $[t_0-1.5$ s, $t_0+1.5$ s] window around the query event and samples at 30 fps. This impractical oracle is expected to perform the best and to set an upper bound for the other practical schemes. The two other dynamic selection schemes use the Hessian score discussed in Sec. 3.2, sampled at 5 fps, to select the best query frame. One Hessian scheme uses full search of post-processed frames up to 1.5 s after the query starts, while the other Hessian scheme uses early-terminated search described in Sec. 3.3 which requires just 0.6 s on average. As can be seen, all three dynamic schemes significantly outperform the static scheme, with the largest gains obtained at large database sizes. The three dynamic schemes also show little differences in recognition accuracy amongst themselves. Hessian-based dynamic selection works nearly as well as the oracle, due to the strong correlation between the number of features and the image-level Hessian score. Early-terminated search also performs comparably with full search, allowing us to substantially reduce the query selection delay from 1.5 s to 0.6 s on average.

5. CONCLUSIONS

We have presented an accurate, low-latency mobile video retrieval system, where a user snaps a photo of a video playing on a TV screen to retrieve and stream the remainder of the video to the mobile device. Visual bookmarks for the user's favorite scenes in movies and TV shows can be automatically stored on his/her mobile device. For this system, we have designed and implemented dynamic selection of a feature-rich frame from the viewfinder frames in a short temporal window around the exact query instant. Our selection techniques efficiently compute Hessian scores for real-time operation on a mobile device. An early termination optimization has been developed to minimize query latency while ensuring with high probability that excellent selections are made. Dynamic selection is shown to significantly improve recognition accuracy, especially for large video databases.

6. REFERENCES

[1] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *ACM Multimedia Information Retrieval*, October 2008, pp. 427–434.

[2] SnapTell, "Media jacket recognition on mobile devices," http://www.snaptell.com/demos/DemoLarge.htm.

[3] Kooaba, "Product logo recognition on mobile devices," http://www.kooaba.com/technology/labs.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[5] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, May 2006, pp. I: 404–417.

[6] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod, "CHoG: compressed histogram of gradients," in *IEEE Computer Vision and Pattern Recognition*, June 2009, pp. 1–8.

[7] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *IEEE Computer Vision and Pattern Recognition*, June 2006, pp. II: 2161–2168.

[8] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *IEEE Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[9] Slingbox, "Streaming TV content to mobile devices," http://www.slingbox.com/go/iphone.

[10] J. Philbin and A. Zisserman, "University of Oxford video retrieval system," in *International Conference on Content-based Image and Video Retrieval*, July 2008, pp. 565–566.

[11] M. Douze, A. Gaidon, H. Jegou, M. Marszalek, and C. Schmid, "INRIA-LEAR's video copy detection system," in *TRECVID Workshop*, November 2008.

[12] H. Jegou, M. Douze, and C. Schmid, "Packing bag-of-features," in *IEEE International Conference on Computer Vision*, September 2009, pp. 1–8.

[13] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod, "Inverted index compression for scalable image matching," in *IEEE Data Compression Conference*, March 2010, p. 525.