

Scalable feature-based video retrieval for mobile devices*

Marleen Morbee
Ghent University,
TELIN-IPI-IBBT
Ghent, Belgium
mmorbee@telin.ugent.be

Marta Mrak
University of Surrey, Centre for
Communication System
Research
Guildford, UK
m.mrak@surrey.ac.uk

Vladan Velisavljević
Deutsche Telekom
Laboratories
Berlin, Germany
vladan.velisavljevic@telekom.de

Wilfried Philips
Ghent University,
TELIN-IPI-IBBT
Ghent, Belgium
philips@telin.ugent.be

ABSTRACT

We present a novel method for video retrieval on mobile devices. The target scenario is the following: features are extracted from a captured query video at the user side and transmitted to a remote server; then, video retrieval is applied within a stored database at the server side and relevant information about the query is returned to the user. In particular, we focus on the user side and propose a scalable method for video feature extraction and encoding taking into consideration the processing capabilities of the device and available bandwidth. Despite these constraints, the first results show a promising accuracy of retrieval.

Keywords

Content-based Video Retrieval, Scale-Invariant Feature Transform, Mobile Search

1. INTRODUCTION

Nowadays, the functionalities of a personal mobile device include much more than just mobile telephony. Built-in cameras and microphones facilitate the creation of audiovisual content and the on-board CPU allows for a number of applications that deal with these different multi-media types. One popular application is content-based audio retrieval [13], where the user records a song using a mobile device and gets the name of the recorded song and the artist from the server. A similar application has been proposed for images in the CD cover recognition system of [12], where the user captures a photo of a CD cover and retrieves the information related to the CD.

*

With the growth of processing power and available bandwidth, mobile content-based *video retrieval* attracts more attention. We will focus on the following target application: the user captures a video from a screen (e.g. TV, projector, cinema, etc.) using a mobile device and gets the relevant information (the name of the movie, brief description, the temporal position of the sequence within the video, etc.) from the server.

So far, many content-based video retrieval systems have been proposed to solve related problems in a non-mobile setup [14, 1, 7, 11, 2, 8]. These systems are all based on algorithms for video content analysis, where the video is indexed by using visual features. The commonly used visual features can be classified into 2 categories: 1) low-level and 2) high-level. 1) Examples of low-level features are color, shape and textures [2, 8]. These features can be extracted from the raw video data with high confidence and they have been successfully applied in a broad range of retrieval applications [14]. The low-level features can be extended to the temporal domain by adding a corresponding temporal dimension [11]. However, they do not analyze the semantic information in the video data. 2) The second category of visual features describe semantic content, such as objects, activities and events [1]. These features allow for a number of new or enhanced video indexing and retrieval applications (e.g. affective video retrieval [7]). However, the process of extracting semantic features is more complex and often requires user interaction [11].

The acquired knowledge from non-mobile video retrieval can be used in the mobile target scenario described above. However, additional challenges and constraints have to be addressed. First, wireless communication between the mobile phone and the server is more costly than processing on the mobile device [10]. Hence, it is unfeasible to shift the retrieval process entirely to the server side by transmitting unprocessed video data from the mobile device to the server. A key issue in our application is the extraction of a set of visual features from the captured video at the mobile phone that is both concise enough for wireless transmission and sufficiently comprehensive for retrieval at the server. In [5], the authors solve this issue by only capturing an image instead of a video at the mobile device. Obviously, with this feature, a great deal of interesting information is lost. Second, the available bandwidth and computational power of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

mobile devices vary over time. Therefore, it is essential to design a process for extraction and retrieval that is scalable to a varying combination of these constraints.

We present a novel method for scalable extraction and encoding of query video information that allows for a trade-off between computational complexity and transmission bit rate. The method relies on robust low-level feature extraction from video sequences based on the scale-invariant feature transform (SIFT) [9]. SIFT features have a number of characteristics that are suitable for mobile video retrieval. First, they are robust to changes in illumination, noise, and minor changes in viewpoint. Furthermore, they are highly distinctive and allow for an easy matching against a (large) database of local features. Finally, SIFT feature matching algorithms have proved to be robust against significantly compressed features [3].

The paper is organized as follows. In Section 2, we describe in more detail our video retrieval solution, and in particular the novel scalable query video representation. In Section 3, we show the performance of our video retrieval method for a test case scenario. Finally, a conclusion and future work are presented in Section 4.

2. VIDEO RETRIEVAL FOR MOBILE DEVICES

Interaction between a mobile phone and a remote server takes place as depicted in Fig. 1. A query is initiated by sending information of the query object to the server through a wireless connection. The amount of information can range from a collection of intra-compressed video frames to a collection of feature descriptors extracted from the query video. The scaling depends on the computational capabilities of the mobile device and the available bandwidth, as explained in Section 2.1. Then, retrieval is applied at the server by matching to the videos in the database and, if a correct match is found, the server returns the required information. The retrieval issues are addressed in Section 2.2.

2.1 Scalable query video representation

At the mobile device, we extract from the captured query video the information needed for retrieval and transmit it to the server. This information is called the *query video representation*. The proposed extraction method provides scalability towards different combinations of processor power and bandwidth, as shown in Fig. 2. Notice that we have focused on scenarios with at least one limitation (in other words, the scenario with high computational power and large bandwidth is not considered), since these scenarios are of particular interest for a mobile setup. Scalability is introduced through three concepts. First, the extraction process can operate in three modes depending on the setup, as explained in the sequel. Second, the captured query video is split into Groups of Pictures (GOP). Each GOP has a size g and consists of one key frame K and its consecutive subordinate frames S^i , with $i = 1 \dots g - 1$ (see Fig. 3). In this work, we assumed a fixed GOP size along the video sequence. Of course, more advanced video indexing methods for key frame selection (leading to variable GOP sizes) [4, 6] would improve the performance of the system, but these are not yet incorporated in this work. Within each mode, the selected size of the GOP g will have an additional in-

fluence on transmission rate and computational complexity: the smaller g , the higher the rate and the more complex the algorithm. Finally, Mode 2 provides a third degree of scalability by allowing a gradual increase of the amount of features and motion vectors, as will be explained in Section 2.1.2.

2.1.1 Mode 1: key frame coding

For mobile phones with very limited computational capabilities, SIFT extraction takes too much time and power. In this case, the key frames from the query video are intra-compressed (e.g. using JPEG with quality factor Q) and sent to the server, and the server performs the feature extraction. The work presented in [5] can be considered as an extreme case of this mode with $g = \infty$ (only one image sent to the server) and without key frame compression. The advantage of this approach is the low computational complexity, but this is at the expense of a lower video matching accuracy and a higher bit rate (as will be discussed in Section 3).

This results in a need for large bandwidth due to the transmission of large data amounts. Information about subordinate frames is discarded, but can be partially recovered at the server by feature matching between two consecutive key frames, which gives a coarse approximation of the SIFT descriptor vector of at least a part of the features in the subordinate frames (see Section 2.2).

2.1.2 Mode 2: feature extraction, tracking and encoding

Advanced mobile devices with embedded processors can perform feature extraction and matching on-board. In this case, we extract from the captured video the features present in the video and track them along the sequence. The query information is then compressed by using intrinsic descriptor statistics [3] and exploiting temporal correlation, that is, sending the motion paths to avoid repetition of the same or very similar features in consecutive frames. Compared to Mode 1, for the same GOP size lower bandwidth is required due to a higher degree of query data compression.

The method is illustrated in Fig. 3. First, SIFT features f_j are extracted from the key frames K_j (j indicates the number of the GOP). Then, the features extracted from consecutive key frames are temporally decorrelated by matching features f_j from a key frame K_j to features f_{j+1} extracted from the subsequent key frame K_{j+1} . Consequently, for matched features only one feature descriptor and the corresponding motion vector (and optionally a residual error between the features) need to be transmitted. Additionally, assuming a linear motion path, the difference in position between a feature f_j in key frame K_j and its match f_{j+1} in key frame K_{j+1} allows us to obtain a coarse approximation \hat{f}_j^i of the descriptor vectors (including position) of the features f_j^i in the subordinate frames S_j^i , with $i = 1, \dots, g - 1$. Notice that this can be done at the server side without a need for transmission of features from subordinate frames (see Section 2.2). The video representation obtained until this point of the algorithm is called the *basic* Mode 2 representation.

In the next (optional) step of the scalable scheme, this *basic* representation is enhanced. False feature matches between K_j and K_{j+1} are removed and the descriptor vector (including position) of the features \hat{f}_j^i is refined. This is done by block matching within a search window I between a block centered on the key point position of f_j in K_j and a block

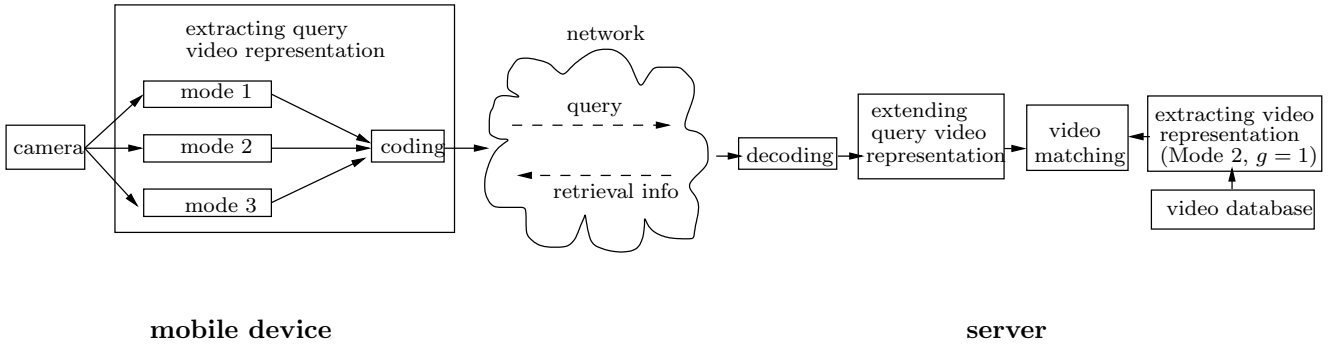


Figure 1: Block diagram for video retrieval with mobile devices.

centered around the estimated position in the subordinate frames S_j^i ($1 \leq i \leq g-1$) (see Fig. 3). If the Minimum Mean Square Error (MMSE) of all the matched blocks is above a threshold, the feature match is discarded. If not, then the position of the block that yields the MMSE is chosen to be the updated position. Optionally, the residual error between the feature f_j and the feature on the updated position can be calculated and encoded. In the last (optional) step, the amount of features can be increased by (partial) feature extraction on the subordinate frames S_j^i ($1 \leq i \leq g-1$). Gradually extracting more features in the subordinate frames and tracking them along the GOP converges finally to a representation that corresponds to the use of a GOP size $g = 1$, where for each frame the features are extracted and matched to a subsequent frame.

2.1.3 Mode 3: key frame feature extraction and encoding

For the key frames SIFT features are calculated, compressed [3] and sent to the server. As in Mode 1, information about subordinate frames is discarded, but can be partially recovered at the server by feature matching between two consecutive key frames (see Section 2.2). For the same GOP size, transmission rate is higher than for Mode 2, since for this mode compression can only be achieved by compressing the feature descriptors and not by exploiting temporal correlation. On the other hand, the computational load at the device is lower than for Mode 2, since only key frame feature extraction is performed and no temporal information is extracted. Compared to Mode 1, more on-board CPU power is needed because of SIFT extraction on the device. In return, lower transmission bandwidth is required for the compressed key frame features than for the intra-compressed key frames.

2.2 Database search

Using the query video representation received from the mobile phone, the video fragment captured in the query video is searched in the video database at the server. The method is depicted in Fig. 1. The database contains for each entry the video representation of the database video extracted within Mode 2 (presented in Section 2.1.2) for $g = 1$.

In case of Mode 1, basic Mode 2 and Mode 3, the query video representation is first extended. In particular, when a video representation of Mode 1 is received, the features are extracted for the intra-decompressed key frames and feature

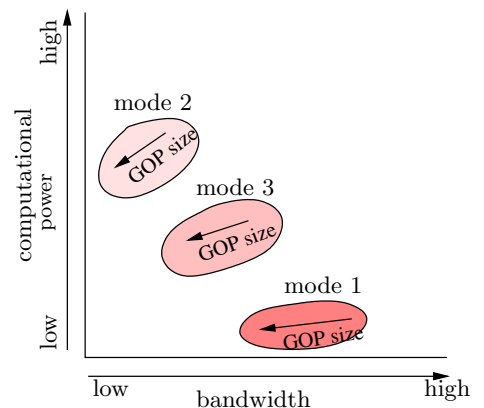


Figure 2: Schematic relationship between required device CPU and transmission bandwidth for the three modes. The arrows point towards larger GOP sizes.

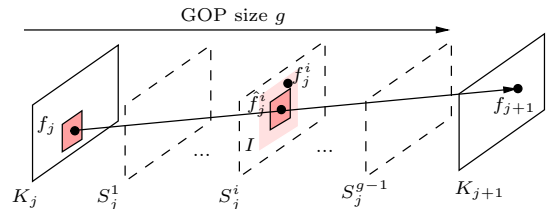


Figure 3: Feature matching and tracking.

matching is done between the features of two consecutive key frames. In case of Mode 3, matching is done between the features of two consecutive key frames. Then, for all the three modes Mode 1, basic Mode 2 and Mode 3, the available feature matches between two consecutive key frames are used to estimate part of the features from the subordinate frames by assuming a linear motion path (as in Fig. 3).

Then, video fragment matching is done by an exhaustive search through the database. More efficient search algorithms exist [6, 4], but in this work we used this basic search strategy for a proof of concept. Let us denote by Q the query video (frame rate f_Q , number of frames N_Q) and by D_i the videos in the database (frame rate f_{D_i} , number of frames N_{D_i}), with $i = 1, \dots, d$ and d the number of videos in the database. In order to perform video matching, we consider for each database video D_i a number of candidate fragments. Each candidate fragment $F_{D_i}^n$ is characterized by the name of its database video D_i and by the frame number of its first frame n , with $0 \leq n \leq N_{D_i} - N_Q - 1$. A candidate fragment has the same number of frames N_Q as the query video, and those N_Q frames correspond to frames in the database video on the time instants $t + \frac{k-1}{f_Q}$, with $k = 1, \dots, N_Q$ and $t = n/f_Q$. During the search, we aim at finding from all the fragments $F_{D_i}^n$ (with possible $n = 0, \dots, N_{D_i} - N_Q - 1$) of all the videos in the database D_i (with $i = 1, \dots, d$), the fragment that is captured in the query video. The database video recorded in the query video is denoted by D_0 and the starting frame of this fragment is n_0 . Then, for each candidate fragment $F_{D_i}^n$, we calculate the amount of matches $m_k(F_{D_i}^n, Q)$ between the features of a k^{th} frame of the query video Q and the features of a corresponding k^{th} frame of a database video fragment $F_{D_i}^n$, with $k = 1, \dots, N_Q$. As matching measure $M_{D_i}(n)$, we use the average amount of feature matches $m_k(F_{D_i}^n, Q)$ per frame, i.e

$$M_{D_i}(n) = \frac{1}{N_Q} \sum_{k=1}^{N_Q} m_k(F_{D_i}^n, Q) \quad (1)$$

The fragment that yields the highest $M_{D_i}(n)$ is then chosen to be the searched fragment $F_{D_0}^{n_0}$

$$F_{D_0}^{n_0} = \arg \max_{D_i, n} M_{D_i}(n). \quad (2)$$

The value of $M_{D_0}(n_0)$ is considered as a measure for the video matching accuracy. Indeed, the higher $M_{D_0}(n_0)$, the higher the probability of a correct video match.

3. EXPERIMENTAL RESULTS

We simulated a real-life setup to evaluate our method. In this experiment, we captured a set of query videos from a LCD screen with a handy camera. In total, we recorded 32 fragments of four test sequences (*Parkjoy*, *Crowd Run*, *Ducks Take Off* and *Old Town Cross*) shown on the screen and recorded under different lightning conditions and from different distances and viewing angles.

The resolution of the query videos is 480×270 . The frame rate of the captured query videos (f_Q) is 25 fps. The number of frames of the query video (N_Q) is between 25 and 50. Two examples of a frame of a query video and its corresponding frame in the database video are shown in Fig. 4. The query videos were matched against a database of 20 videos, with

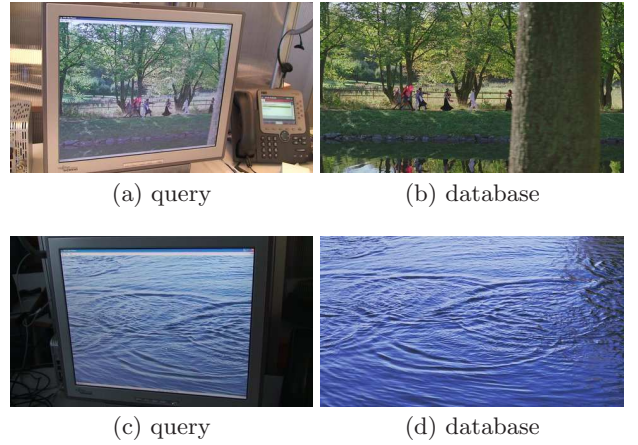
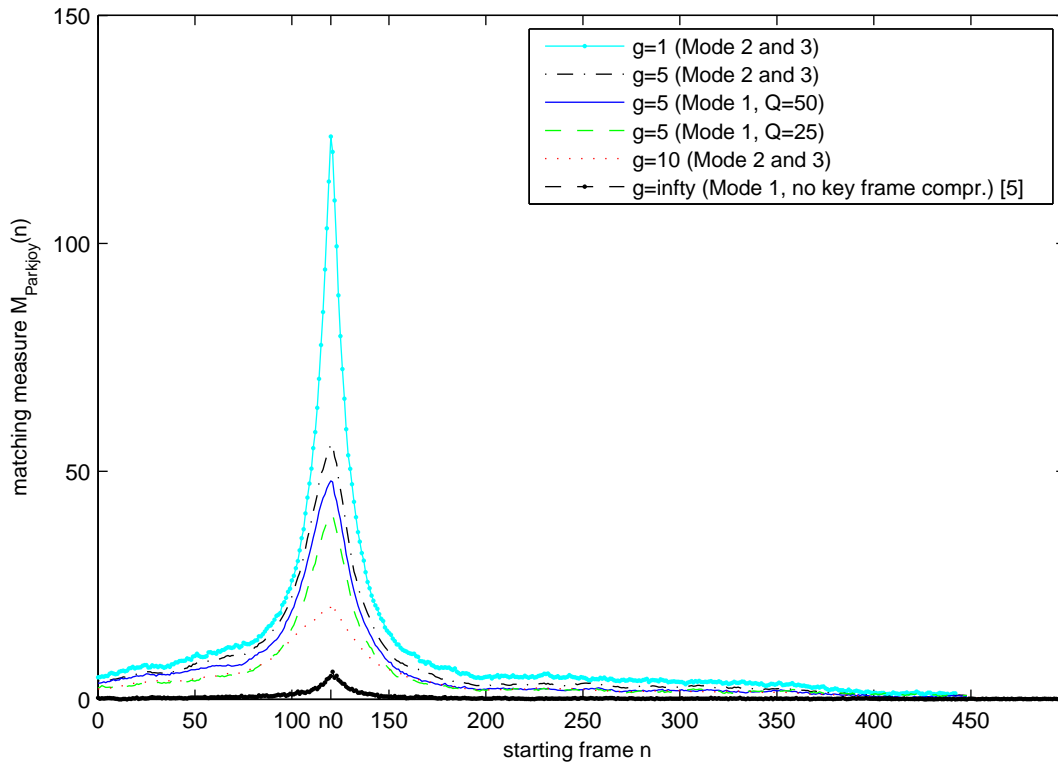


Figure 4: Query video vs. database video, for the sequences *Parkjoy* (above) and *Ducks Take Off* (below).

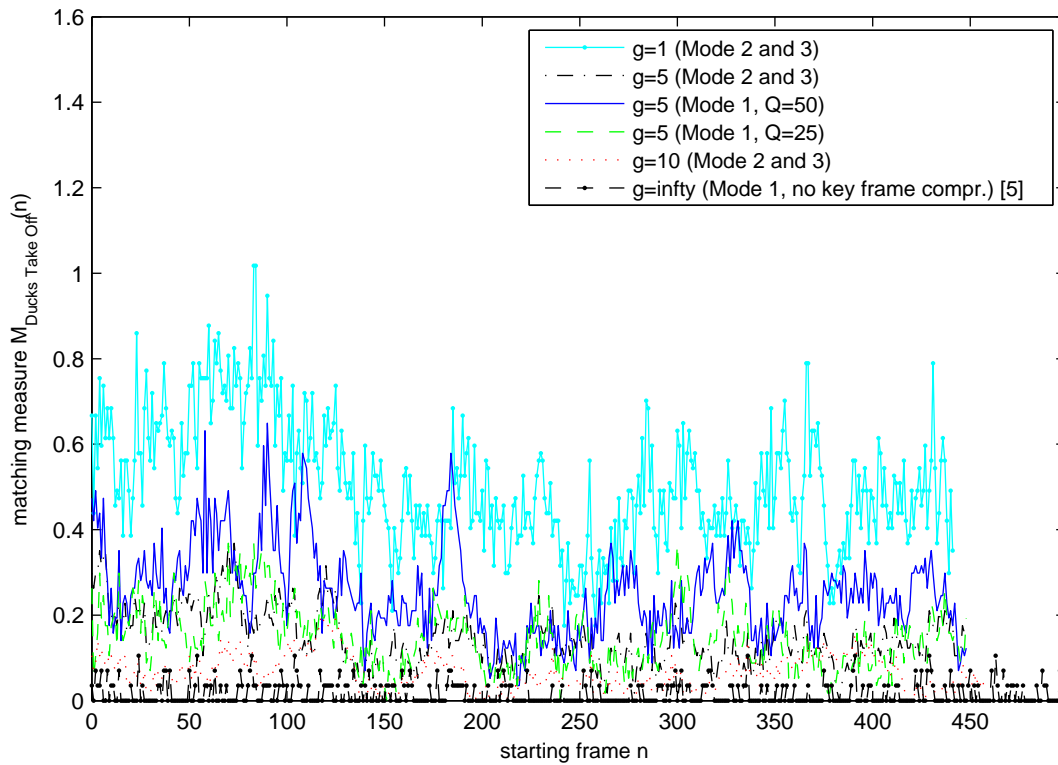
a total of 40 minutes playing time. The frame rate of the database videos (f_{D_i} , $i = 1, \dots, 20$) is 50 fps. We evaluated our algorithm in terms of video matching accuracy and bit rate. For the intra-compression of the key frames in Mode 1, we used JPEG compression with quality factor Q ($1 \leq Q \leq 100$). For the SIFT feature extraction the threshold on key point contrast was set to 0.04 and the threshold on ratio of principal curvatures was set to 10. The feature descriptors were transform coded with the method proposed in [3].

3.1 Video matching accuracy

As described in Section 2.2, our goal is to retrieve from all the videos in the database the video fragment that corresponds to the captured query video. In Fig. 5(a), we plot the resulting video matching measure $M_{Parkjoy}(n)$ (see Eq. (1)) versus the starting frame number n for a query of *Parkjoy* (with $n_0 = 120$, $N_Q = 28$) matched to the fragments of the same database video entry *Parkjoy*. Similarly, the same query is matched to another database entry *Ducks Take Off* and the corresponding matching measure $M_{Ducks Take Off}(n)$ is shown in Fig. 5(b). The five curves show the results for different modes and GOP sizes g . For Mode 2, we consider the basic video representation obtained without the optional motion and descriptor vector refinement. In this case, the video matching measures for Mode 2 and Mode 3 are equal. Note that, however, the rate for the two modes is different, as will be discussed in Section 3.2. For basic Mode 2 and Mode 3, the results are shown for $g = 1$ (cyan solid line with points), $g = 5$ (black dash-dotted line) and $g = 10$ (red dotted line). For Mode 1, we plotted the results for $g = 5$ with $Q = 50$ (blue solid line) and $Q = 25$ (green dashed line). For Mode 1, we also plotted the results for the extreme case $g = \infty$ and without key frame compression, which corresponds to the query video representation used in [5] (black dashed line with points). Fig. 5 shows that we obtain very accurate video matching results. First, observing the difference in matching measure between $M_{Parkjoy}(n)$ (Fig. 5(a)) and $M_{Parkjoy}(n)$ (Fig. 5(b)), it is clear that in the query video a fragment from the database entry *Parkjoy* was captured. Second, in Fig. 5(a), the well-defined maximum of $M_{Parkjoy}(n)$ at $n = n_0 = 120$ shows that exact



(a) *Parkjoy* query matched to *Parkjoy* database



(b) *Parkjoy* query matched to *Ducks Take Off* database

Figure 5: Illustrative example of the video matching accuracy. Matching measure $M_{Parkjoy}(n)$ and $M_{Ducks Take Off}(n)$ for a query of *Parkjoy* with as starting frame of the fragment $n_0 = 120$. We observe a clear peak on $n_0 = 120$ in Figure (a) (matching database video) and a very low matching measure in Figure (b) (non-matching database video). The correct fragment is retrieved with a temporal accuracy within one frame.

video fragment retrieval with a temporal accuracy within 1 frame is achieved. Observe that, the smaller g , the sharper the peak, or in other words, the better the video matching accuracy $M_{Parkjoy}(n_0 = 120)$. This is logical, since for a larger g there are fewer key frames, which finally leads to a lower amount of features (and matches with the database fragments) per frame. A second reason is that the amount of matches between features in key frames that are further apart due to a larger g will be lower, so that less information about features in subordinate frames can be estimated. Also, if we compare the three graphs with $g = 5$, we can see that the JPEG compression applied before feature extraction in Mode 1 reduces the video matching accuracy compared to Mode 2 and Mode 3. This is caused by the JPEG artifacts that introduce feature distortions, loss of features and false features during the feature extraction at the server. Obviously, the higher the quality factor Q , the lower this negative impact. We also observe that the video matching accuracy in the case of the video representation of [5] (Mode 1, $g = \infty$, without key frame compression) is significantly lower than for the other cases. This is due to the fact that in this video representation only one key frame of the query video is included and hence, no time information is considered. Finally, notice that in Fig. 5(b) the average amount of mismatches between the non-matching database video entry *Ducks Take Off* and the query video of *Parkjoy* is very small, so as desired very low values for the matching measure are obtained. In this figure, the average amount of mismatches is slightly larger for a smaller GOP size. This is because in general query video representations with smaller GOP size contain more features, which leads to a higher probability of a false match.

Very similar results are obtained for the other queries, also for query videos of sequences that are little distinctive over time, e.g. moving waves in the water of *Ducks Take Off* (see Fig. 4).

3.2 Bit rate

In Fig. 6, we show how the video matching accuracy for a query of *Crowd Run* ($N_Q = 38$), $M_{Crowd Run}(n_0)$, relates to the rate spent on the query video representation, for the three modes described in Section 2.1. We compare the results with the video representation used in [5] (Mode 1, $g = \infty$ and without key frame compression).

For Mode 1, JPEG compression with quality $Q = 25$ and $Q = 50$ is shown. The black dotted line corresponds to Mode 1 ($Q = 50$), the green dashed line to Mode 1 ($Q = 25$), the red dash-dotted line to Mode 3, and the cyan solid line to Mode 2. The different points (indicated by x-marks) in the graph correspond to the different GOP sizes $g = 1, \dots, 10$. The blue triangle corresponds to [5] (Mode 1, $g = \infty$ and without key frame compression). Notice that operating points are desired to lie as close as possible to the top left corner of the graph. Notice that, for the same video matching accuracy, Mode 2 provides a more efficient rate than Mode 3 since for compression not only the intrinsic descriptor statistics are exploited but also the temporal correlation. Mode 1 is least rate-efficient because the transmission of JPEG compressed images requires more bits than sending only the compressed feature descriptors. Moreover, for the same GOP size, the video matching accuracy in Mode 1 is lower. This is caused by the JPEG artifacts that distort the feature extraction at the server, as explained in

Section 3.1. The video representation of [5] (extreme case of Mode 1) does not perform well due to a lack of key frame compression and a low corresponding video matching accuracy because of the large GOP size ($g = \infty$) (as explained in Section 3.1).

Notice that, in contrast to Mode 1 and Mode 3, the scalability of the representation in Mode 2 allows each point on the curve (in between the points for the different GOP sizes) to be achieved by an adequate motion and descriptor vector refinement and additional feature extraction from the subordinate frames. The video representation of [5] (extreme case of Mode 1) does not provide any scalability, so there is only one operating point possible.

We observe that, for the three modes, larger GOP sizes in the video representations yield smaller rates. This behavior is caused by a large portion of the bit rate being spent for encoding the compressed features of the key frames with no temporal correlation to the features in the neighboring key frames. Therefore, the more key frames, the larger the rate, since the rate saved by more temporal decorrelation for smaller GOP sizes does not compensate for the larger amount of transmitted key frame features. The results for the other query videos are similar.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel scalable method for video retrieval on mobile devices. The target scenario was the following: the user captures with his phone a video of a movie shown on a screen. The mobile device communicates information about the query video to the server, where video matching against a video database takes place. Information about the retrieved movie is then returned to the user. We focused in particular on the mobile device side and proposed a scalable method for the extraction and encoding of a query video representation, that allows for accurate video matching. The method relies on robust SIFT feature extraction to overcome the distortions in the query video. Three modes are supported, and each mode complies with a different combination of on-board CPU power and available transmission bandwidth. First results from a simulated real-life setup showed a promising video matching accuracy with 100% video recognition rate for the considered database and a temporal accuracy within one frame.

Future work includes the incorporation of a large video database with efficient hierarchical search algorithms and the use of feature motion paths as an additional clue in difficult video matching scenarios. Also, we will incorporate an appropriate key frame selection algorithm to determine the optimal GOP size at each time instant. Furthermore, we will investigate more thoroughly how the length of the query video and the number of extracted and tracked SIFT features relate to the retrieval accuracy, bit rate and device processing power, in order to optimally allocate resources in a real-life environment. Finally, we plan to implement and test the algorithm on mobile devices in a real-time setup.

5. REFERENCES

- [1] S. Adali, K. S. Candan, K. S. C. S. shing Chen, K. Erol, and V. S. Subrahmanian. Advanced video information system: Data structures and query processing. *Multimedia Systems*, 4:172–186, 1996.

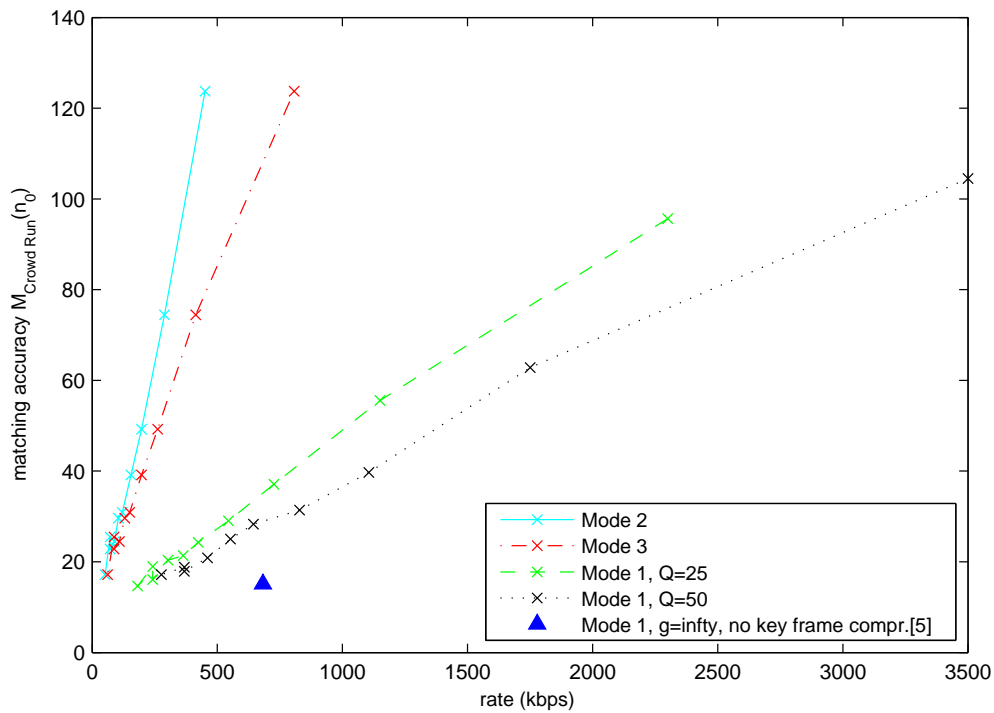


Figure 6: Video matching accuracy $M_{Crowd Run}(n_0)$ vs. bit rate for a query of *Crowd Run*.

- [2] A. Y. Aslandogan and C. T. Yu. Techniques and systems for image and video retrieval. *IEEE Trans. on Knowl. and Data Eng.*, 11(1):56–63, 1999.
- [3] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, J. Singh, and B. Girod. Transform coding of image feature descriptors. In *Visual Communications and Image Processing*, San Jose, CA, USA, 2009.
- [4] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269 – 1279, 1999. Content-based retrievals;Key frame extractions;Key frame hierarchy;Systematic algorithms;.
- [5] C.-M. Chen, Y.-Z. Wang, H.-A. Wang, and C.-Y. Chiu. Digital video retrieval via mobile devices. In *ESCIENCE '08: Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pages 376–377, Washington, DC, USA, 2008. IEEE Computer Society.
- [6] S. Dagtas, W. Al-Khatib, A. Ghafoor, and R. L. Kashyap. Models for motion-based video indexing and retrieval. *IEEE Transactions on Image Processing*, 9(1):88 – 101, 2000. Content based retrieval;Motion modeling;Video database;Video indexing;.
- [7] A. Hanjalic and L.-Q. Xu. Affective video content representation and modeling. *IEEE Transactions on Multimedia*, 7(1):143–154, 2005.
- [8] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, February 2006.
- [9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
- [10] M. Murphy. Content based image retrieval on mobile devices. In *Talk presented at the 2008 GSRC Annual Symposium.*, September 2008.
- [11] M. Petkovic and W. Jonker. Integrated use of different content derivation techniques within a multimedia database management system. *Journal of Visual Communication and Image Representation*, 15(3):303 – 329, 2004. Multimedia Database Management Systems.
- [12] S. S. Tsai, D. Chen, J. P. Singh, and B. Girod. Rate-efficient, real-time cd cover recognition on a camera-phone. In *ACM Multimedia*, Vancouver, BC, Canada, 2008.
- [13] E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE MultiMedia*, 3(3):27–36, 1996.
- [14] H. Zhang, J. Wu, D. Zhong, and S. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30:643–658, 1997.